UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Bachelor's thesis

Study, design, construction and validation of a flight director for VFR flights for en-route operation of a Cessna C-152

Bachelor's degree in Aerospace Technology Engineering

Student:
OLIVELLA I MARTÍ, Guillem

Director:
PÉREZ LLERA, Luis Manuel

Delivery date: June 10th, 2019

Polytechnic University of Catalonia

To my grandfather,
for being my first engineering teacher.

Contents

List of Figures **4**

List of Tables **6**

Glossary **7**

1 Acknowledgements **9**

2 Introduction **10**
2.1 Abstract . . . 10
2.2 Declaration of honor . . . 11
2.3 Aim . . . 12
2.4 Scope . . . 12
2.5 Requirements . . . 13
2.6 Justification . . . 13

3 Background and state of the art **14**
3.1 Instrument definition: flight director . . . 14
3.1.1 Commercial solutions developed and design trends . . . 15
3.1.2 Analysis of alternatives . . . 18

4 Flight director development **19**
4.1 Instrument operational frame . . . 19
4.2 General approach: Flight director goal . . . 21
4.3 Physical model development . . . 24
4.3.1 Reference frames . . . 24
4.3.2 Flight mechanics development . . . 25
4.4 3D approach: Navigation analysis . . . 34
4.5 Automation analysis: control action . . . 36
4.5.1 Coupled PID controllers . . . 37
4.5.2 Linear-quadratic regulator . . . 37
4.5.3 Model predictive control . . . 38
4.5.4 Development of the final solution . . . 38
4.6 iOS application development . . . 45
4.6.1 Sensors used and data treatment . . . 45
4.6.2 Graphic interface . . . 46
4.6.3 Application algorithm . . . 49

5 Results and Validation: Flight tests **51**
5.1 Flight test 1 . . . 53
5.2 Flight test 2 . . . 54
5.3 Flight test 3 . . . 56
5.4 Flight test 4 . . . 58

6 Discussion and Conclusions **60**
6.1 Design evaluation and results analysis . . . 60
6.2 Environmental impact . . . 61
6.3 Security evaluation . . . 61

7 Future improvements **62**
7.1 Future work programming . . . 64

Bibliography **65**

Appendix A Flight path Simulink simulation **67**
A.1 Full Model Hierarchy . . . 67
A.2 Model variables . . . 68
A.2.1 Navigational, control and aircraft parameters . . . 68
A.3 Generic Model Block Diagram . . . 69
A.4 Model/Pilot . . . 70
A.5 Model/Airplane Model . . . 71
A.6 Model/Kinematic Model . . . 72
A.7 Model/Waypoint Selector . . . 72
A.8 Model/Waypoint Selector/ CalculateNextWaypoint . . . 73
A.9 Flight Director . . . 74
A.10 Model/Flight Director/Angle Converter . . . 75
A.11 Model/Flight Director/Roll Limiter . . . 75
A.12 Model/Flight Director/Theta Limiter . . . 76
A.13 Model/Flight Director/Subsystem 1 . . . 77
A.14 Model/Flight Director/Subsystem 2 . . . 77

Appendix B iOS application files **78**
B.1 Graphical User Interface . . . 78
B.2 Control action file . . . 82

List of Figures

3.1 Simplified scheme of a flight director 14
3.2 Block diagram of the aircraft control loop without a FD 15
3.3 Block diagram of the aircraft control loop with a simple FD 16

4.1 Diagrams of routes with different types of waypoints and legs over the same geographical positions 20
4.2 Diagram of the instantaneous velocity when the aircraft has the optimum attitude to cross the waypoint at the desired altitude. 21
4.3 Diagram of the instantaneous velocity and its projections when the aircraft is not in the same altitude as the waypoint. 22
4.4 Carrot-chasing algorithm 23
4.5 Non-linear guidance law 24
4.6 Analytic models and experimental results 32
4.7 Pairs of maximum roll and bank angles with the security factor 32
4.8 Pitch and roll commands for different K 33
4.9 Generic spheric traingle with its 6 characteristic angles 34
4.10 Scheme of the generic case where the aircraft path is not aligned to the waypoint direction. 35
4.11 Schemes of the two spherical triangles under analysis. 35
4.12 Scheme of two coupled PID with dependable saturation 37
4.13 Control action block diagram 38
4.14 Simulation of the aircraft horizontal path, vertical profile and attitude commands for 3 consecutive waypoints. 42
4.15 Simulation of the aircraft horizontal path for different headings between hypothetical waypoints 43
4.16 Simulation of the aircraft horizontal path for different wind conditions 43
4.17 Simulation of the aircraft horizontal path for different aerodynamic velocities . . . 43
4.18 Simulation of the aircraft vertical path for different vertical currents of air 44
4.19 Simulation of the aircraft vertical path for different velocities 44
4.20 Experimental data of course and altitude 46
4.21 First view of the application developed 47
4.22 Second view of the application developed 48
4.23 Design of the movable parts of the artificial horizon 48

5.1 Photography of the flight director installed in the airplane 51
5.2 Horizontal and vertical paths of Flight Test 1. 53
5.3 Horizontal and vertical paths of Flight Test 2. 54

5.4 Horizontal and vertical paths of Flight Test 3. 56
5.5 Horizontal and vertical paths of Flight Test 4. 58

7.1 Second method to detect when to change the target point. 64

A.1 Generic Simulink Block Diagram 69
A.2 Pilot Simulink Block Diagram 70
A.3 Airplane Model Simulink Block Diagram 71
A.4 Kinematic Model Simulink Block Diagram 72
A.5 Waypoint selector Simulink Block Diagram 72
A.6 Flight Director Simulink Block Diagram 74
A.7 Subsystem 1 Simulink Block Diagram 77
A.8 Subsystem 2 Simulink Block Diagram 77

List of Tables

3.1 Comparative table of the three hardware alternatives. 18

4.1 Flight indications for airspeed and engine RPM 30
4.2 Experimental data to determine maximum attitude positions 30
4.3 PID parameters for vertical and horizontal control. 40

5.1 In-flight check list for co-pilot. 52

7.1 Data to be improved and possible ways to do so 63
7.2 Prediction of hours needed to improve the project by doing certain tasks 64

A.1 Pilot reaction time numerical data 70

Glossary

AMSL	Above Mean Sea Level.
C_L	Lift coefficient.
C_{L0}	Lift coefficient at zero angle of attack.
$C_{L\alpha}$	Lift coefficient slope with respect to angle of attack.
C_D	Drag coefficient.
C_{D0}	Drag coefficient at zero lift coefficient.
D	Airplane drag.
$\vec{F}$	Resultant force acting on the aircraft.
$\vec{F_A}$	Aircraft aerodynamic forces.
$\vec{F_W}$	Aircraft weight.
g	Gravity.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
$\dot{h}$	Aircraft vertical velocity.
k	Drag quadratic coefficient.
K	Proportion between altitude and course errors.
K_p	Proportional PID parameter for course.
K_i	Integral PID parameter for course.
K_d	Derivative PID parameter for course.
K_{pA}	Proportional PID parameter for altitude.
K_{iA}	Integral PID parameter for altitude.
K_{dA}	Derivative PID parameter for altitude.
L	Airplane lift.
L_{wh}	Euler transformation matrix from the local horizon frame to the wind frame.
m	Total mass of the aircraft.
M	Mach number.
N	Derivative PID secondary parameter for altitude.
O_h	Origin of the local horizon reference frame.
p_w	x component of the angular velocity of the airplane referenced from the wind basis.
P	Airplane position.
P_e	Aircraft engine power.
P_p	Aircraft propeller power.
q	Cross-track point.
q_w	y component of the angular velocity of the airplane referenced from the wind basis.
Q	Aerodynamic lateral force.
r_w	z component of the angular velocity of the airplane referenced from the wind basis.

Re	Reynolds number.
R_e	Earth Radius.
s	Virtual target point.
s'	Incorrect virtual target point.
S	Aircraft wing surface.
t	Time.
T	Magnitude of the thrust vector.
T_s	Time step.
V	Magnitude of the aerodynamic velocity.
$\vec{V}$	Aircraft aerodynamic velocity.
$\vec{V_a}$	Wind velocity with respect to the ground.
$\vec{V_g}$	Aircraft velocity with respect to the ground.
$\vec{V_v}$	Vertical velocity with respect to the ground.
W	Next waypoint position.
W_i	Previous waypoint position in a flight leg.
W_{i+1}	Following waypoint position in a flight leg.
W_z	Altitude of the following waypoint referenced to the aircraft altitude.
x_h	x component of the local horizon reference frame.
x_w	x component of the wind reference frame.
y_h	y component of the local horizon reference frame.
y_w	y component of the wind reference frame.
z_h	z component of the local horizon reference frame.
z_w	z component of the wind reference frame.
α	Angle of attack.
γ	Flight path angle.
δ_1	First Euler angle of rotation.
δ_2	Second Euler angle of rotation.
δ_3	Third Euler angle of rotation.
ϵ	Thrust angle of attack. Angle formed by velocity and thrust.
ζ	Angle between the horizontal velocity and the position of W with respect to the airplane.
η_p	Propeller - engine power transmission efficiency.
θ	Aircraft pitch angle.
κ	Angle between thrust and the x component of the body reference frame.
λ	Geographical longitude.
μ	Velocity bank angle.
π	Thrust level position.
ρ	Air density.
φ	Geographical latitude.
χ	Velocity heading angle.
χ_w	Wind velocity angle.
ψ	Angle between the velocity and the position of s with respect to the airplane.
$\vec{\omega}$	Angular velocity of the airplane referenced from the wind basis.

1 | Acknowledgements

I would like to express my deep gratitude to Professor Luis Manuel Pérez Llera for his useful and constructive recommendations on this project. Special thanks should also be given to Marc Sabaté and Jaume Olivella for taking part in the project as flight test pilots.

2 | Introduction

2.1 Abstract

A flight director for a Cessna 152 (single engine, fix propeller aircraft) for en-route operation has been designed, implemented and tested. The hardware, totally independent of the aircraft instruments and systems, was an iOS platform with GNSS capabilities (iPhone XR). The path following law employed has been a pure pursuit algorithm in the horizontal path; and, the flight technique used consisted on reaching the desired altitude at the same time that the aircraft was established in the correct course to cross the target point with levelled wings. The algorithm used involved two parallel PIDs controllers (one for the horizontal path and the other for the vertical profile) with variable, interconnected saturations to display pitch and bank commands to the pilot. The parameters that characterized the control action have been tuned using heuristic methods during flight tests and using a Simulink model, through which the aircraft-pilot-instrument interaction has been simulated. The flight tests executed by three different subjects showed that although following the flight director commands resulted in crossing the waypoints at the correct altitude, the workload for the pilot was excessive. The stress showed by the pilots when following commands of pitch, roll and engine regime by the flight director was bigger than when not following the instrument. Moreover, the comparison of the flights profiles between the different pilots with different quantities of flight hours revelled that a good understanding of the aircraft dynamics was fundamental to efficiently following the instrument commands.

2.3 Aim

The aim of this project is to study the development of a flight director, the flight mechanics needed for the prototype, the algorithm and the limitations of the result. More precisely, the flight director will provide guidance for en-route operation of a Cessna 152 with VFR. The pre-flight input variables will be GNSS points with information about geographical latitude and longitude and altitude AMSL that will be treated as fly-over waypoints. During the flight, the flight director will receive information about one or more built-in gyroscopes, accelerometers and GNSS information totally independent from the aircraft systems and avionics.

2.4 Scope

- **Study of the flight director requirements, inputs and outputs of the system.**

  A general approach to the problem will be done and all the variables of the problem will be defined and studied considering the requirements of the desired design. The inputs and outputs of the system will be studied and their limitations will be determined.

- **Study of the state of the art of flight directors for commercial and general aviation.**

  A general research involving flight directors and navigational systems will be performed. And the state of the art will be determined.

- **Flight mechanics study: physical and mathematical approach.**

  The problem will be defined from a physical and mechanical point of view. Important equations of the system will be determined.

- **Dynamical system study and control design.**

  The system will be studied from a control point of view. The best control system will be determined.

- **Algorithm development of the flight director.**

  The algorithm of the system will be developed in the chosen programming language.

- **Flight tests with a C152 and iterative improvements.**

  Flight tests with different pilots will be done, improvements will be done as a function of the results obtained. The process will be iterative.

- **Analysis of the data obtained.**

  A final flight test will be performed and the data obtained will be evaluated to analyze the final result.

- **Final evaluation and study of future improvements.**

  A final evaluation of the system will be done considering many factors as the final result, evolution of the prototype, comments of the pilots, etc.

- **Redaction of a report explaining both: the theoretical approach and the experimental results.**

  Finally, a report will be written with all the information considered as important. Other pieces of information that could be considered as results of the project such as videos of the flight tests will be displayed.

2.5 Requirements

- The flight director has to be totally independent from the aircraft systems.
- The flight director will be designed to help the pilot for navigation and performance purposes but in no case will be an instrument to rely on.
- Safety measures will be taken for the flight tests.

2.6 Justification

A flight director is a device that provides information to the pilot about the pitch and roll attitude necessary to perform a specific manoeuvre. Nowadays, these instruments can be found in advanced systems like Garmin G1000 or incorporated in PFD in commercial aviation: for example, in Airbus or Boeing families. However, there are no flight directors for simple airplanes independent from the aircraft avionics. When flying in VFR, the pilot has to maintain constant visual contact with the terrain, constantly checking the actual and predicted position and speed of the airplane. A VFR flight director for general navigation would solve problems related with the early phases of practical learning; situations of low visibility and assistance for the pilot in new routes. This project will be a first approximation to the design of a system for a C-152 that will resolve the stated problems. Since the flight director will be designed for VFR flights, it will not be validated for approximation, landing and taking-off phases of flight. The system will be programmed to make the pilot fly-over a predefined sequence of waypoints at a predefined altitude. Other manoeuvres such as simulating a VOR radial interception will not be considered in this project.

3 | Background and state of the art

The instrument characteristics will be introduced and the design trends will be discussed. Regarding the state of the art, different flight directors from different aircrafts will be presented as well as their operational properties. Finally, an analysis of different alternatives will be done in order to discuss which hardware will be used to develop de instrument.

3.1 Instrument definition: flight director

As stated previously, a flight director is a device usually built together with an artificial horizon of an airplane designed to provide information to the pilot about the attitude of the aircraft necessary to perform a specific manoeuvre or follow a pre-designed route [1]. This information can be displayed in many ways, but a typical one is to show two independent magenta bars (one horizontal and the other vertical) that the pilot has to match with the airplane symbol in the artificial horizon. The instrument can be understood as a device that synthesizes data derived from many sources in order to reduce the demands of attitude, altitude and heading to a demand of maintaining only attitude [2].

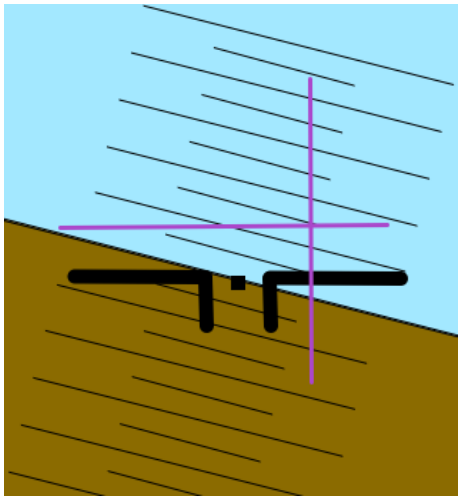


Figure 3.1: Simplified scheme of a flight director

Figure 3.1 shows a scheme of typical flight director. In this case, the airplane is rolling to the left; but the pilot has to roll the airplane to the right and point the aircraft's nose up to fly the desired route. This is: he has to make the airplane symbol coincide with the purple cross. Nowadays, flight directors are installed as integrated instruments of the airplane, since the inputs usually include indicated airspeed, altitude and radio navigational information among others.

Many modern commercial aircrafts (including Airbus and Boeing families) have a flight director installed at least in one of their glass cockpit screens; business jets and general aviation aircrafts (such as Cirrus SR-22 and Cessna 172) can have the instrument installed, too. Concerning general aviation, one of the most typical integrated systems that includes a flight director is Garmin G1000. The Garmin G1000 Pilot's Guide for Cessna Nav III defines the flight director operation as follows: *"The flight director function provides pitch and roll commands to the AFCS and displays them on the PFD. With the flight director activated, the aircraft can be hand-flown to follow the path shown by the Command Bars. Maximum commanded pitch (+20° /-15°) and roll (22°) angles, vertical acceleration, and roll rate are limited to values established during AFCS certification. The flight director also provides commands to the autopilot."* [3]

3.1.1 Commercial solutions developed and design trends

The first approach to a flight director was the Sperry Zero Reader flight director (1948) which proved to be a useful instrument in variable weather conditions [4]. From the beginnings of aviation, there has been an effort to simplify instrumentation by combining many indicators in a single display and, usually, no new information has been added. All the attempts to integrate the instruments in a panel have reduced the workload for the pilot but in all cases he still needed to interpret and assimilate the information to make an estimation of the control action required. To understand how flight director general design trends have evolved, a block diagram of the traditional aircraft control loop without a FD installed is presented in Figure 3.2 [2].

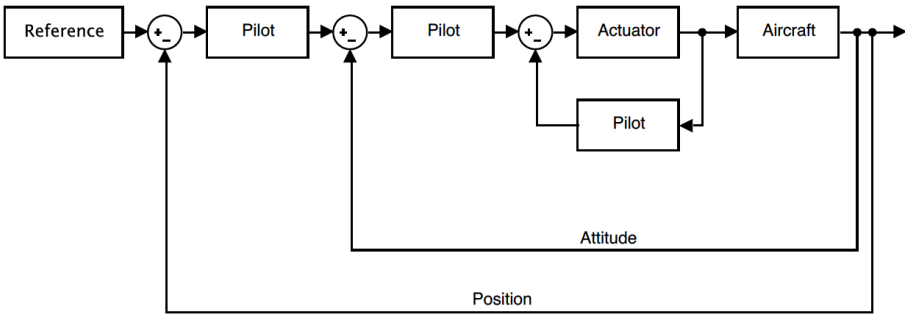


Figure 3.2: Block diagram of the aircraft control loop without a FD

As it can be seen, precision when sensing flight data and navigational parameters and effectiveness in computing the control action needed is vital to accomplish an optimal flight path. Along the years, accuracy of the sensors has been improved as well as the way sensors outputs have been presented in the cockpit [5]. A flight director is used to compute the required control action to determine the correct attitude of the aircraft as a function of the instruments outputs. Both interpretation and computation of the response is done by the flight director computer. At this point, one could think that the next step would be to substitute the information displayed as attitude for information related to the control stick position by including another appropriate transfer function in accordance to the aircraft dynamics. However, a stick demand presentation would imply a high frequency signal that would be translated to an increase in the pilot workload [2]. The aircraft control loop with a simple flight director included is represented with the block diagram of Figure 3.3.

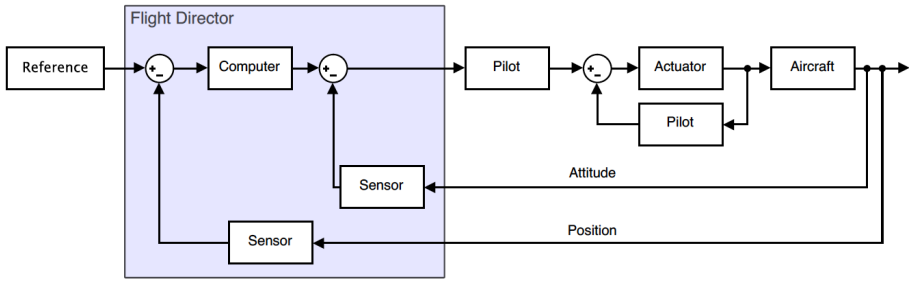


Figure 3.3: Block diagram of the aircraft control loop with a simple FD

Nowadays, more complex systems and flight modes have been developed. The state of the art of this system can be studied analysing the modes of operation of two advanced systems: the Boeing and Airbus families (representing the aircrafts certified under CS25 with performance Class A) and Garmin G1000 (usually used for aircrafts certified under CS23 with performance Class B). Both systems consist of control loops separated into vertical and horizontal modes that can be activated independently. Generally, flight directors have two type of modes category: simple (which consists on maintaining a predefined parameter as heading or altitude) or advanced (which consists on more complex control loops that can represent higher-order systems) [6].

The following flight director modes correspond to the Airbus A320 and Boeing 737 aircrafts, used as examples to represent the Airbus and Boeing families [7] [8].

For the vertical modes:

- ALT (for AIRBUS) and ALT (for BOEING): Aircraft holds selected altitude. [Simple mode]
- V/S (for AIRBUS) and V/S (for BOEING): Aircraft will maintain a constant vertical speed. [Advanced mode]
- CLB/DES (for AIRBUS) and VNAV (for BOEING): Aircraft will climb or descent in accordance to a vertical, predefined flight plan. [Advanced mode]
- OP CLB/DES (for AIRBUS) and LVL CHG (for BOEING): Aircraft will climb and maximum thrust and descend at idle. [Advanced mode]
- APPR (for AIRBUS) and APP (for BOEING): Aircraft will follow a vertical 3D approach based on available guidance. [Advanced mode]

For the horizontal modes:

- HDG (for AIRBUS) and HDG (for BOEING): Aircraft will hold a selected heading. [Simple mode]
- NAV (for AIRBUS) and LNAV (for BOEING): Aircraft will follow the legs predefined in the flight plan. [Advanced mode]
- LOC (for AIRBUS) and VOR LOC (for BOEING): Aircraft will establish on a localizer. [Advanced mode]
- APPR (for AIRBUS) and APP (for BOEING): Aircraft will follow a lateral guidance available on a 3D approach. [Advanced mode]

On the other hand, the Garmin G1000 flight director can be studied [3]. When activated, the instrument can also operate independently in any Lateral - Vertical mode pair chosen by the pilot.

For the vertical modes:

- (default - PIT): Aircraft holds aircraft pitch attitude. [Simple mode]
- ALT: Aircraft holds current altitude. [Simple mode]
- VS: Aircraft holds vertical speed. [Advanced mode]
- FLC: Aircraft holds airspeed while climbing or descending to a selected altitude. [Advanced mode]
- VNV: Aircraft captures and tracks descent legs of an active vertical profile. [Advanced mode]
- APR: Aircraft captures and holds glideslope. [Advanced mode]
- GA: Aircrafts holds constant pitch angle. [Advanced mode]

For the horizontal modes:

- (default - ROL): Aircraft holds the current roll attitude or rolls the wings depending on the commanded bank angle. [Simple mode]
- HDG: Aircraft captures and tracks the selected heading. [Simple mode]
- NAV: Aircraft captures and tracks the selected navigation source (GPS, VOR, LOC). [Advanced mode]
- APR: Aircarft captures and tracks the selected navigation source for landing (GPS, VOR, LOC). [Advanced mode]
- GA: Aircraft hold wings level for go around. [Advanced mode]

As it has been explained, the pilot can select a lateral and a vertical flight control independently. However, in cruise phases of flight it is common to select the Navigation lateral control and Altitude Hold or Vertical Path Tracking vertical control in order to make the airplane fly a predesigned route. After analysing the lateral and vertical modes, it can be concluded that the three types of flight directors presented (Airbus A320, Boeing B737 and Garmin G1000) are very similar although small differences can be found in their operation.

In terms of regulations, there are not important rules regarding the operation of a flight director. And, in terms of usage, it is always recommended by the operator to follow the flight director if it is activated; otherwise, it should be turned off so as to avoid any possible confusion [3].

3.1.2 Analysis of alternatives

The hardware used to develop the flight director for the Cessna C-152 has to be decided before designing the control action because the available sensors and their characteristics depend of the physical interface used. Since one of the project requirements is to create a flight director independent of the aircraft systems, the following platforms have been evaluated:

1. Arduino with GNSS capability.
2. Raspberry Pi with GNSS capability.
3. iOS platforms (such as iPad or iPhone).

Has to be said that the three options could be chosen for the development of the flight director and as a consequence, pros and cons have been evaluated for all of them: regarding the iOS option, the location data is already filtered and the inputs from the sensors are fused through an already implemented complex algorithm. In second place, iOS CPU capabilities are higher than the ones of the two other options, which can be used to improve the program performance. On the other hand, Arduino and Raspberry Pi environments are easier to program than an iOS app (since a native application has to be developed). In order to compare all the different important characteristics of the three alternatives, a comparative table has been constructed (see Table 3.1). In this table, different punctuations have been given to the different specifications: a 5 corresponds to a more favourable characteristic for the flight director and a 1, a less favourable one (2, 3 and 4 have been used for intermediate punctuations). After considering the implications of the characteristics of the three systems, the iOS app option has been chosen.

	iOS platform (iPhone app)	**Arduino UNO**	**Raspberry PI 3**
CPU capabilities	5	1	3
User-friendly interface	5	1	2
Capability of extending the final product	5	1	1
Data internal treatment	5	3	1
GNSS precision	3	5	5
Programming complexity	1	5	3
Hardware design complexity	5	2	1
TOTAL:	**29**	**18**	**16**

Table 3.1: Comparative table of the three hardware alternatives.

4 | Flight director development

The design of the instrument will consist on many phases which will be discussed in a sequential order to create an optimal response of attitude correction. The final objective of the development process is to identify the best control action for the system under analysis. To do so, a physical model will be proposed, as well as a mathematical model to deal with the navigational problems derived from the, assumed, rounded Earth. Finally, the control action will be tuned and the response of the system will be discussed for many types of perturbations.

4.1 Instrument operational frame

The airplane for which the flight director will be developed is a Cessna C-152, flying under VFR rules and following a VFR Operational Routing. A Cessna C-152 is a two-seat, single engine aircraft with a wingspan of 10.2 m, and a wing area of 14.9 m^2. Its empty weight is 490 kg and it has a maximum take-off weight of 757 kg. The power plant used for the flight tests will be a two-blade fixed pitch propeller mounted on a Lycoming O-235 engine. Regarding its performance characteristics, the most important ones are [9]:

Maximum speed at sea level ..... 110 knots
Cruise, 75% Power at 8,000 ft ..... 107 knots
75% Power at 8,000 ft ..... 350 NM
Rate of climb at sea level ..... 715 FPM
Stall Speed (Flaps up) ..... 48 knots
Stall Speed (Flaps down) ..... 43 knots
Never Exceed Speed ..... 145 knots

Although the flight director development does not depend on the flight rules under which the airplane will fly, it is important to state that the regulations that will be considered when developing the flight tests will be Visual Flight Rules. These rules require both: to maintain constant visual contact to terrain and to fly in conditions better or equal than VMC (Visual Meteorological conditions). In the EU, VMC are described by EASA in the document *Standardized European Rules of the Air* and are summarized in the following points [10]:

1. When not flying above congested areas, the minimum altitude is 500 ft above any obstacle within 500 ft.

2. When flying above congested areas, the minimum altitude is 1000 ft above the highest fixed object within 600 m of the aircraft.

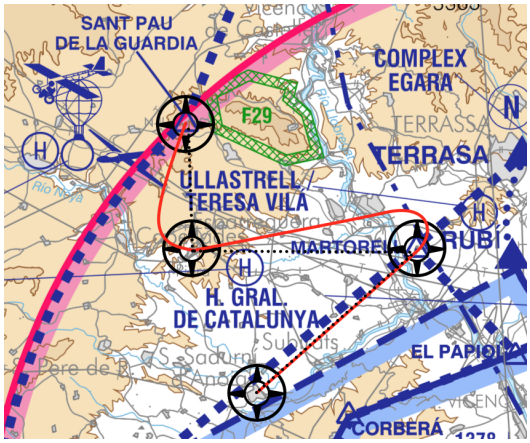
3. Aircraft can only operate in Airspace classes B, C, D, E, F, G.

4. The pilot has to maintain a distance from any cloud bigger than: 1 500 m horizontally and 300 m (1 000 ft) vertically.

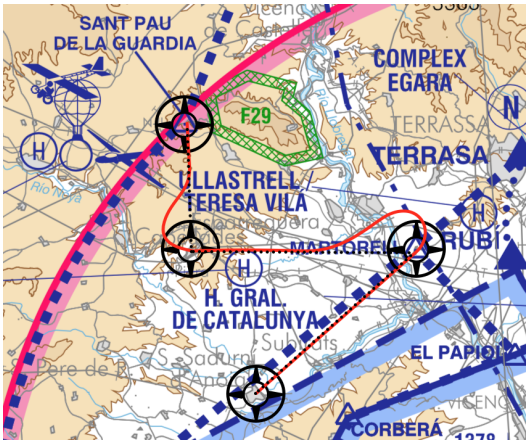
5. At and above 3 050 m (10 000 ft) AMSL, the minimum flight visibility has to be: 8 km.

6. Below 3 050 m (10 000 ft) AMSL, the minimum flight visibility has to be: 5 km.

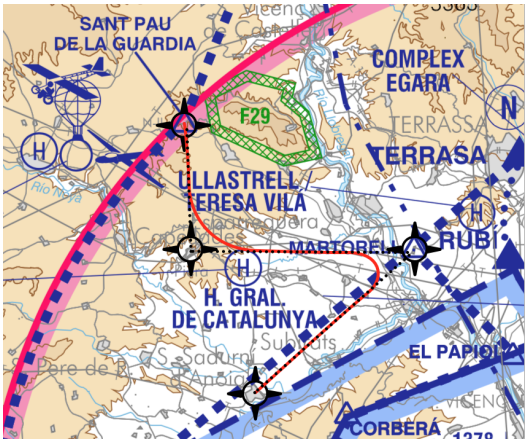
When flying in such conditions, pilots present a VFR operational flight plan which is a document explaining the waypoints that will be followed during the flight. It is mainly composed by a table where the waypoints are sorted and the tracks and headings necessary to fly the desired route are calculated. The time between waypoints is also written as well as the different altitudes that will be flown. When flying in VFR, it is common to use the geographical references described in the routing as fly-over waypoints; these are en-route points that have to be overflown before changing the heading to go to the next waypoint. This type of route is not the most efficient one since the condition of over-flying points is usually a time consuming restriction. Despite this fact, in VFR flights, it is common to fly following this consideration since it is a good way to monitor the exact position of the aircraft and its speed.



(a) Route with fly-over waypoints with direct-to-fix legs



(b) Route with fly-over waypoints with track-to-fix legs



(c) Route with fly-by waypoints

Figure 4.1: Diagrams of routes with different types of waypoints and legs over the same geographical positions

When considering this type of route, two options can be followed once the waypoint is crossed: to fly a direct-to-fix leg (which consists on going directly to the next waypoint following the shortest distance - see figure 4.1a) or to fly a track-to-fix leg (which consists on joining, as soon as possible the track connecting the previous and next waypoint - see figure 4.1b). On the other hand, if the waypoints are not overflown, they are called fly-by waypoints and the route followed is, consequently, shorter that the two mentioned previously - see figure 4.1c. Figure 4.1 shows the three types of routes discussed considering the following waypoints: S. Sadurní d'Anoia (N 41.43°, E 1.79°), Martorell (N 41.48°, E 1.92°), Capellades (N 41.53°, E 1.68°) and Sant Pau de la Guardia (N 41.61°, E 1.75°) at an arbitrary non-defined altitude [11].

4.2 General approach: Flight director goal

As it has been explained, a flight director is a device that instructs the pilot on how to fly a predetermined trajectory. In our case, the airplane will fly a trajectory such that it crosses all the checkpoints at a specified altitude. When an operational flight plan is followed in VFR (considering fly-over waypoints), it is common to climb to the altitude of the next waypoint and change the heading of the airplane as soon as the previous checkpoint is crossed. After this, a straight and leveled flight is maintained until the next waypoint is crossed and new instructions are considered. The control system of the airplane is the modification of pitch and roll, which together, change the rate of climb and rate of turn of the airplane. If the kinematics of the problem are analysed and the instantaneous velocity and the position of the airplane are taken into account, it can be concluded that the flight director has two simultaneous goals:

- Maintain the airplane at the same altitude as the next waypoint.
- Align the instantaneous velocity vector with the position of the next waypoint referenced from the airplane.

The fact of aligning the velocity with the following point is a path-following algorithm called pure pursuit algorithm and it is the most common method used in routes with fly-over waypoints with direct-to-fix legs [12]. If both conditions mentioned above are met simultaneously, the aircraft will cross the waypoint at the desired altitude. This situation is depicted in figure 4.2.

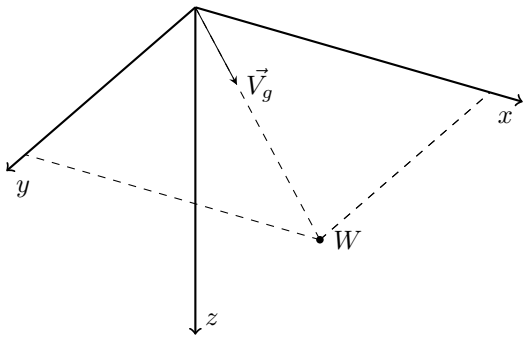


Figure 4.2: Diagram of the instantaneous velocity when the aircraft has the optimum attitude to cross the waypoint at the desired altitude.

However, in an arbitrary situation in which the altitude of the airplane and the waypoint are not the same and the instantaneous velocity is not aligned with the position of the waypoint referenced from the aircraft, the flight director has to display pitch and roll values such that they:

- Reduce the angle formed by the projection of the instantaneous velocity to the horizontal plane of the local horizon and the projection of the vector formed by the next waypoint and the aircraft (the origin of the local horizon reference frame). This angle will be called ζ and it will be referenced to the local horizon.
- Apply an optimum rate of climb or rate of descent if the altitudes of the airplane and the waypoint do not coincide. The projection of the velocity vector to the z-axis of the local horizon will be called $\vec{V_v}$.

This situation can be seen in figure 4.3.

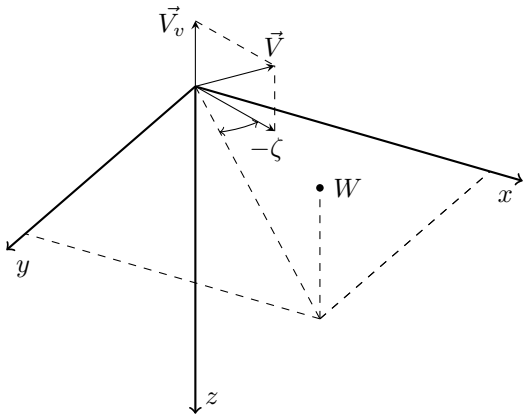


Figure 4.3: Diagram of the instantaneous velocity and its projections when the aircraft is not in the same altitude as the waypoint.

Mathematically, the problem is reduced to find an optimal control action for the flight director in order to display, in the flight director:

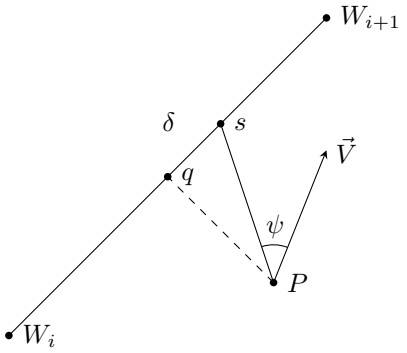
$$\{\theta \text{ and } \mu\} : \begin{cases} \dfrac{d \mid \zeta \mid}{dt} = 0 & \text{if } \zeta = 0 \\ \dfrac{d \mid \zeta \mid}{dt} < 0 & \text{if } \zeta \neq 0 \\ \dfrac{d \mid W_z \mid}{dt} = 0 & \text{if } W_z = 0 \\ \dfrac{d \mid W_z \mid}{dt} < 0 & \text{if } W_z \neq 0 \end{cases} \tag{4.2.1}$$

The values of $\dot{\zeta}$ and $\dot{W_z}$ have to be selected taking into account the aerodynamics of the aircraft, pilotage laws and physical limitations of the system.

Has to be said that the condition of aligning the velocity vector with the line joining the next waypoint and aircraft position is an arbitrary decision that will make the pilot fly a specific trajectory (fly-over waypoints with direct-to-fix legs). However, other possibilities could have been considered for the horizontal path [13]. Two alternatives will be explained, although neither of them will be developed in detail in this project: Carrot-chasing algorithm and non-linear guidance law.

Carrot-chasing algorithm

The Carrot-chasing algorithm is a geometric path-following technique used to create track-to-fix trajectories (see figure 4.4a). It consists on computing the cross-track point (q), which is the nearest point of the line joining the previous and next waypoint to the airplane (P). After this, a virtual target point VTP (s) is computed, which is a point on the leg joining the two waypoints at a distance δ from q (always located nearer to the second waypoint). Then, an angle ψ is calculated between the velocity vector and the line joining s and p. The airplane, then, has to be flown in a way that reduces the angle ψ more and more. This can be done using any controller. The only parameter that has to be tuned to use this algorithm is δ. The smaller δ, the quicker the aircraft joins to the leg between the two waypoints. On the other hand, if δ is very big, the aircraft trajectory will point more and more to the destination.



(a) Carrot-chasing algorithm diagram

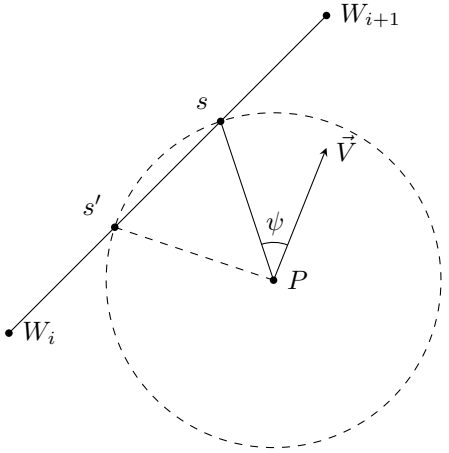
while *not above last waypoint* **do**
  if *above next waypoint* **then**
    Change next waypoint;
  else
    Determine point q;
    Update location of VTP s;
    Update ψ;
    Reduce ψ;
  end
end

(b) Algorithm to follow a straight line

Figure 4.4: Carrot-chasing algorithm

Non-linear guidance law

The non-linear guidance law algorithm also is a geometric path-following method used to create track-to-fix trajectories (see figure 4.5). It also consists on finding and following a virtual target point: In this case, however, the VTP is found by the intersection between the leg (joining the two waypoints) and a circle of constant radius centred on the aircraft position. By computing the intersection, three main things can happen: the circle intersects the leg exactly in one point, the circle intersects it in two points or the circle does not intersect it. If the circle intersects in two points, the nearest one to the next waypoint is selected. If the circle does not intersect, the VTP selected is the point of the leg nearest to the airplane (the cross-track point q). Once s is found, ψ is computed and, as explained above, any controller can be used to reduce it. The parameter that has to be tuned in this case is the circle radius: the smaller the radius, the quicker the airplane joins the leg track. In contrast, the bigger the parameter, the more directly the airplane arrives to the destination waypoint [14].



(a) Non-linear guidance law diagram

while *not above last waypoint* **do**
  if *above next waypoint* **then**
    Change next waypoint;
  else
    Determine point s and s';
    Select correct point s;
    Update ψ;
    Reduce ψ;
  end
end

(b) Algorithm to follow a straight line

Figure 4.5: Non-linear guidance law

4.3 Physical model development

Once we have the mathematical problem identified, it is time to study the physical model to estimate its limitations and select the best control action and tune it.

4.3.1 Reference frames

To develop the equations to study the general dynamics of the aircraft and to define the navigational problem, some reference frames have to be defined [15]:

- Navigational or Earth reference frame
- Local horizon reference frame
- Wind reference frame

Navigational Reference frame The origin will be defined as an arbitrary point on the surface of the Earth. In our problem, it will be defined as the reference point of the departure airport. The x-axis will be a unitary vector pointing the geographical north pole and the z-axis will point to the center of the Earth. The y-axis will be the unitary vector necessary to complete a positive orthonormal basis.

Local horizon reference frame The local horizon frame will be defined as follows: O_h - position of the aircraft, x_h - pointing to the north, z_h - pointing to the center of the Earth, y_h - vector necessary to form an orthonormal positive basis.

Wind reference frame The wind reference frame will be referenced to the local horizon and it will be obtained as rotation using three Euler angles. These angles will be applied using a 321 convention to orientate the aircraft velocity in wind axes:

- **Velocity heading angle** ($\delta_3 = \chi$). Which refers to the angle formed by the reference direction x_h(north) and the projection of the aerodynamic velocity to the horizontal plane of the local horizon.
- **Flight path angle** ($\delta_2 = \gamma$). Which refers to the angle formed by the aerodynamic velocity and its projection to the horizontal plane.
- **Velocity bank angle** ($\delta_1 = \mu$). Which is the angle formed by the second component of the wind frame (y_w) and the intersection of the horizontal plane with the plane formed by the second and third component of the wind frame ($y_w - z_w$).

Applying this angles to the Euler transformation matrix with a 321 convention, it can be obtained that the transformation matrix from the local horizon to the wind frame is:

$$L_{wh} = \begin{pmatrix} \cos\gamma\cos\chi & \cos\gamma\sin\chi & -\sin\gamma \\ \sin\mu\sin\gamma\cos\chi - \cos\mu\sin\chi & \sin\mu\sin\gamma\sin\chi + \cos\mu\cos\chi & \sin\mu\cos\gamma \\ \cos\mu\sin\gamma\cos\chi + \sin\mu\sin\chi & \cos\mu\sin\gamma\sin\chi - \sin\mu\cos\chi & \cos\mu\cos\gamma \end{pmatrix} \tag{4.3.1}$$

4.3.2 Flight mechanics development

The linear momentum theory states that:

$$\vec{F} = m \cdot \frac{d\vec{V}}{dt} \tag{4.3.2}$$

Given that the airplane is a rotatory system, the linear momentum equation has to be formulated as:

$$\vec{F} = m(\frac{\partial\vec{V}}{\partial t} + \vec{\omega} \times \vec{V}) \tag{4.3.3}$$

where:

$$\vec{\omega} = \begin{pmatrix} p_w \\ q_w \\ r_w \end{pmatrix} \quad \text{and} \quad \vec{V} = \begin{pmatrix} V \\ 0 \\ 0 \end{pmatrix}$$

$\vec{\omega}$ is the angular velocity of the airplane referenced from the wind basis. $\vec{V}$ is the aerodynamic velocity of the airplane expressed from the wind reference frame; which is implicitly defined from the definition of the wind basis. We will assume that the external force vector is formed by three main factors: Aircraft weight ($\vec{F_W}$), Aerodynamic forces ($\vec{F_A}$) and thrust from the engine($\vec{F_T}$). The aircraft weight is known from the local horizon reference frame (mg $\vec{z_h}$); consequently:

$$(\vec{F_W})_w = L_{wh} \times (\vec{F_W})_h = L_{wh} \times \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} -mg\sin\gamma \\ mg\cos\gamma\sin\mu \\ mg\cos\gamma\cos\mu \end{pmatrix} \tag{4.3.4}$$

The aircraft aerodynamic forces, in the wind reference frame is:

$$(\vec{F_A})_w = \begin{pmatrix} -D \\ -Q \\ -L \end{pmatrix} \tag{4.3.5}$$

However, it will be assumed that the pilot flies in a coordinated way, which means that there are no lateral aerodynamic forces: Q = 0. Finally, thrust can be expressed using the aircraft reference frame as a force exerted longitudinally. As an approximation, it will be assumed that

the thrust force is exerted in the same plane as the $z_w - x_w$, which is true if the pilot performs a coordinated flight. In such case, the thrust angle of attach (ε) will be defined as the angle formed by the velocity and thrust vectors:

$$(\vec{F_T})_w = \begin{pmatrix} T\cos\varepsilon \\ 0 \\ T\sin\varepsilon \end{pmatrix} \tag{4.3.6}$$

Regrouping equations 4.3.3, 4.3.4, 4.3.5, 4.3.6, the following is obtained:

Given that the rotation of the three angles (γ, μ and χ) have been done in three different basis, their derivative can also be expressed in three different basis:

$$\dot{\vec{\chi}}_h = \begin{pmatrix} 0 \\ 0 \\ \dot{\chi} \end{pmatrix} \qquad \dot{\vec{\gamma}}_3 = \begin{pmatrix} 0 \\ \dot{\gamma} \\ 0 \end{pmatrix} \qquad \dot{\vec{\mu}}_2 = \begin{pmatrix} \dot{\mu} \\ 0 \\ 0 \end{pmatrix}$$

Doing the necessary rotations, the following relations can be obtained:

$$\dot{\vec{\chi}}_w = (L)_{w2} \cdot (L)_{23} \cdot (L)_{3h} \cdot \dot{\vec{\chi}}_h$$

$$\dot{\vec{\gamma}}_w = (L)_{w2} \cdot (L)_{23} \cdot \dot{\vec{\gamma}}_3$$

$$\dot{\vec{\mu}}_w = (L)_{w2} \cdot \dot{\vec{\mu}}_2$$

Developing and substituting:

$$\vec{\omega} = \dot{\vec{\chi}}_w + \dot{\vec{\gamma}}_w + \dot{\vec{\mu}}_w = \begin{pmatrix} -\sin\gamma \\ \cos\gamma\sin\mu \\ \cos\gamma\cos\mu \end{pmatrix} \dot{\chi} + \begin{pmatrix} 0 \\ \cos\mu \\ -\sin\mu \end{pmatrix} \dot{\gamma} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \dot{\mu} \tag{4.3.7}$$

Finally, regrouping equations 4.3.3, 4.3.4, 4.3.5, 4.3.6, 4.3.7 the following is obtained:

$$\begin{aligned} T\cos\varepsilon - D - mg\sin\gamma - m\dot{V} &= 0 \\ mg\cos\gamma\sin\mu + mV(\dot{\gamma}\sin\mu - \dot{\chi}\cos\gamma\cos\mu) &= 0 \\ -T\sin\varepsilon - L + mg\cos\gamma\cos\mu + mV(\dot{\gamma}\cos\mu + \dot{\chi}\cos\gamma\sin\mu) &= 0 \end{aligned} \tag{4.3.8}$$

Given that the aircraft structure is fixed and known, $\alpha - \varepsilon$ is constant over time; this means that ε can be written as a function of the angle of attack and a parameter dependent on the geometry of the airplane:

$$\varepsilon = \alpha + \kappa$$

In the same way, the flight path angle (γ) has been defined as the angle between the aerodynamic velocity vector and the horizontal plane. The angle of attack (α) is the angle formed between the chord line of the wing and the aerodynamic velocity vector. This means that the angle formed between the chord line of the wing and the local horizon (aircraft pitch angle: θ) can be defined as:

$$\theta = \alpha + \gamma \tag{4.3.9}$$

The aerodynamic and propulsive forces are also know; lift and drag forces can be modelled as follows:

$$\begin{aligned} L &= \frac{1}{2}\rho V^2 \cdot S \cdot C_L(\alpha, Re, M) \\ D &= \frac{1}{2}\rho V^2 \cdot S \cdot C_D(\alpha, Re, M) \end{aligned} \tag{4.3.10}$$

Since Cessna C-152 is propeller-driven aircraft, the produced power has to be modelled as:

$$\begin{aligned} P_p &= T \cdot V \\ P_p &= P_e(\pi) \cdot \eta_p(V, N) \end{aligned} \tag{4.3.11}$$

The propeller efficiency depends on the velocity of the aircraft and the angular velocity of the propeller (N). Since the aircraft is not fitted with a constant-speed propeller, the efficiency does not depend on the blades pitch. Where π is the engine thrust level.

From a kinematic point of view, 3 equations have to be added to the system in order to define the motion of the aircraft from the local horizon point of view. A rotational transformation will be done in order to obtain the desired relations from the wind reference frame. Moreover, a $\vec{V_a}$ is added to the $\vec{V_w}$ to represent the air velocity from the local horizon.

$$\vec{V_h} = L_{hw}\vec{V_w} + \vec{V_a} = L_{hw}^T\vec{V_w} + \vec{V_a} \tag{4.3.12}$$

The L_{hw} matrix will be obtained from the theory developed above as well as the $\vec{V_w}$ vector that has been defined as $\vec{V}$:

$$\begin{pmatrix} \dot{x_h} \\ \dot{y_h} \\ \dot{z_h} \end{pmatrix} = \vec{V_h} = L_{hw}^T \begin{pmatrix} V \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} V_{ax} \\ V_{ay} \\ V_{az} \end{pmatrix} = \begin{pmatrix} V\cos\gamma\cos\chi + V_{ax} \\ V\cos\gamma\sin\chi + V_{ay} \\ -V\sin\gamma + V_{az} \end{pmatrix} \tag{4.3.13}$$

Rearranging all the equations discussed since now, the following non-linear system of equations can be obtained:

$$\begin{aligned} &\textbf{Dynamic equations} \\ &T\cos\varepsilon - D - mg\sin\gamma - m\dot{V} = 0 \\ &mg\cos\gamma\sin\mu + mV(\dot{\gamma}\sin\mu - \dot{\chi}\cos\gamma\cos\mu) = 0 \\ &-T\sin\varepsilon - L + mg\cos\gamma\cos\mu + mV(\dot{\gamma}\cos\mu + \dot{\chi}\cos\gamma\sin\mu) = 0 \\ &\textbf{Aerodynamic equations} \\ &L = \frac{1}{2}\rho V^2 \cdot S \cdot C_L(\alpha, Re, M) \\ &D = \frac{1}{2}\rho V^2 \cdot S \cdot C_D(\alpha, Re, M) \\ &\textbf{Propulsive equations} \\ &P_p = T \cdot V \\ &P_p = P_e(\pi) \cdot \eta_p(V, N) \\ &\textbf{Kinematic equations} \\ &\dot{x_h} = V\cos\gamma\cos\chi + V_{ax} \\ &\dot{y_h} = V\cos\gamma\sin\chi + V_{ay} \\ &\dot{z_h} = -V\sin\gamma + V_{az} \\ &\textbf{Other relations} \\ &\varepsilon = \alpha + \kappa \\ &\theta = \alpha + \gamma \end{aligned} \tag{4.3.14}$$

A set of 12 equations describe the motion of the model proposed from a kinematic and dynamic point of view. However, in order to proceed, some approximations will be done and the simplified system and its limitations will be analysed.

The flight will be considered at a constant Reynolds and at a very low Mach number. The aerodynamic equations will be simplified without losing much precision in the results obtained. Since a C-152 flies at a TAS of 65-100 kts, at an altitude of 3000 ft with a standard atmosphere, it has a Mach number that varies from 0.10 to 0.15. This shows that the flow around the aircraft can be considered incompressible. Moreover, since the velocity range is small, the Reynolds can be considered constant around the entire aircraft:

$$\begin{aligned} C_L(\alpha, Re, M) &\approx C_L(\alpha) \\ C_D(\alpha, Re, M) &\approx C_D(\alpha) \end{aligned} \tag{4.3.15}$$

The thrust will be considered aligned with the airflow. As it has been seen, ε depends on the angle of attack (α) and a constant parameter defined by the aircraft geometry κ. The angle of attack varies throughout the flight in a range of, more or less, 10°; however, κ is constant. This means that, since $\varepsilon = \alpha + \kappa$, the angle ε can also change within a range of 10°. This variable will be considered 0; this means that $\cos\varepsilon \approx 1$ and $\sin\varepsilon \approx 0$: the thrust will act entirely on the x-axis of the wind frame, which is an acceptable hypothesis for the simplified model.

The thrust will only depend on N and V. It has been seen that $P_e(\pi)$. It can also be stated that the angular velocity of the engine blades depends on π and the velocity (V). This means that $N = N(\pi, V)$ where $N(\pi, V)$ is a strictly increasing function in π and therefore, injective in π. Consequently, π can be expressed as a function of V and N ($\pi = \pi(N, V)$); considering this fact: P_e can be written in terms of N and V since:

$$P_e(N, V) = P_e(\pi) \circ \pi(N, V) \tag{4.3.16}$$

Moreover,

$$T(\pi, N, V) = \frac{P_e(\pi) \cdot \eta_p(V, N)}{V} = \frac{P_e(N, V) \cdot \eta_p(N, V)}{V} = T(N, V) \tag{4.3.17}$$

During the flight, the transitional states between the stationary manoeuvres will not be considered. Stationary manoeuvres are considered combinations of climbs and descents at constant rate and coordinated turns and straight-and-level flight. This means that:

$$\dot{V} = \dot{\gamma} = 0 \tag{4.3.18}$$

Applying all these hypothesis to the set of equations 4.3.14 and doing the appropriate substitutions, the following set of equations can be obtained:

$$\begin{aligned} T(V, N) - \frac{1}{2}\rho V^2 S C_D(\alpha) &= mg \cdot \sin(\theta - \alpha) \\ \frac{1}{2}\rho V^2 S C_L(\alpha) &= mg \cdot \frac{\cos(\theta - \alpha)}{\cos\mu} \\ \dot{\chi} &= \frac{g}{V} \cdot \tan\mu \\ \dot{h} = -\dot{z_h} &= V \sin(\theta - \alpha) + V_{az} \end{aligned} \tag{4.3.19}$$

Where we have the following unknowns:

- 2 kinematic: $\dot{h},\ \dot{\chi}$
- 3 pilotage: $\theta,\ N,\ \mu$

- 2 aerodynamic: $V,\ \alpha$

Since we have 7 unknowns and 4 equations, 7 - 4 = 3 degrees of freedom. These 3 degrees of freedom can be read as the 3 pilotage unknowns controllable by the pilot: thrust (N), pitch (θ) and bank angle (μ). If the pilot configures the airplane with a defined set $\{\theta,\ \pi,\ \mu\}$, the airplane will fly in a determined way defined by the set $\{\dot{h},\ \dot{\chi},\ V,\ \alpha\}$.

As it has been explained, a flight director displays a proposed pitch and bank angle to fly a desired flight path (*id est* a desired $\dot{h}, \dot{\chi}$). As a consequence, it is necessary to propose a defined system of equations with two unknowns: θ and μ. To reduce the DOF of the system the following restrictions will be applied:

For safety reasons, the aircraft's aerodynamic velocity will be maintained at a predefined value during a same manoeuvre. The parameter can be maintained constant by the pilot by means of reading the airspeed indicator and controlling the thrust by means of changing the RPMs of the engine.

The two remaining conditions can be interpreted as the control of $\dot{h}$ and $\dot{\chi}$ by the pilot or the flight director These will be related to the two pilotage variables commanded by the flight director (θ and μ) by means of a control algorithm that will be explained below.

The set of equations 4.3.19 can be rewritten as:

$$\begin{aligned} T(V,N) - \frac{1}{2}\rho V^2 S C_D(\alpha) &= mg \cdot \frac{\dot{h}}{V} \\ \frac{1}{2}\rho V^2 S C_L(\alpha) &= mg \cdot \frac{\cos(\theta-\alpha)}{\cos\mu} \\ \dot{\chi} &= \frac{g}{V}\tan\mu \end{aligned} \tag{4.3.20}$$

This new equations imply that $\dot{\chi}$ depend mainly on the bank angle, when $\dot{h}$ is highly related to $T(V,N)$. On the other hand (and analysing the second equation) it can be seen that the aircraft's velocity can be controlled by means of changing the pitch angle. In fact, this is one of the rules applied by pilots when flying: "**Velocity is controlled by pitch and altitude is controlled by thrust**". Remembering one of the restrictions imposed, the flight director cannot be connected to any airplane system (which includes thrust level and airspeed indicator). However, to control the entire dynamics of the aircraft it will be necessary to have an auto-throttle and an airspeed indicator. The system has then two main problems: first, the flight director does not have enough inputs to know the value of the variables that define the state of the system and, secondly, the system does not have control of thrust, which is an important input to control height.

As a consequence to these limitations, the flight director cannot be designed for an entire range of the aircraft thrust level or velocity. The solution proposed is to design a control action that supposes the unknown variables and, according to these estimated values, it generates pitch and roll outputs in order to fly a specific path. The way to do this is to make the pilot control the thrust level in a way that it maintains the engine RPM at a specific value (which can be changed if the flight director is retuned). Then, the system would be able to generate θ and μ outputs that would assure a flight at a specific velocity (which, as it has been said, is a necessary restriction for safety reasons) that can vary from manoeuvre to manoeuvre.

Taking into account, the flight manual of the aircraft and the safe range of velocity and engine regime, it will be proposed to fly following the above stated indications:

	Ascent flight	**Horizontal flight**	**Descent flight**
Airspeed (kts)	70	-	90
Engine RPM	2300	2300	2100

Table 4.1: Flight indications for airspeed and engine RPM

Has to be noted that for horizontal flight, only a restriction has to be given (in this case engine regime), since the other one will be the consequence of the restriction of maintaining altitude. The maximum pitch and roll commands for both the horizontal and descent flight can be considered independent since the trust input will be sufficient to accomplish the indications indicated in table 4.1. However, for ascent flight, the engine may not produce enough thrust to respond to a request of maximum pitch and roll. As a consequence, the relationship between maximum θ and μ have to be found when flying at an airspeed of 70 kts and 2300 RPM. The process that will be followed to find such relationships will consists on finding some values experimentally and tune a some parameters of a model derived from equations 4.3.19 and the following aerodynamic equations:

$$C_L(\alpha) = C_{L0} + C_{L\alpha} \cdot \alpha \qquad C_D(C_L) = C_{D0} + k \cdot C_L^2 \tag{4.3.21}$$

With:

$$C_{L0} = 0.35 \qquad C_{L\alpha} = 4.77$$

The lift coefficient data has been obtained from [16].

Experimentally, data has been collected to find relationships between the parameters under study when flying in the conditions of velocity and RPM stated above, the following has been obtained:

Bank (°) $\pm$ **2°**	**Pitch (°)** $\pm$ **2°**	**Ascent velocity (FPM)** $\pm$ **100 FPM**
0	8	650
7	7	550
12	7	400
18	4	200

Table 4.2: Experimental data to determine maximum attitude positions

As it can be seen, in table 4.2, pitch data is both imprecise and inexact. Then, in order to estimate the relationship between the set of roll angles and pitch angles the following will be done: In first place, an analytic model to find the relationship between ascent velocity and bank angle will be found to validate the results. Afterwards, an analytical solution will be proposed to find the pitch angles that correspond to each ascent velocity.

To determine the analytic relationship between ascent velocity and roll angle, equations 4.3.19 have been rewritten in a different way:

$$\begin{gathered} T(V,N) - \frac{1}{2}\rho V^2 S C_D(\alpha) = mg \cdot \sin(\theta - \alpha) \\ \frac{1}{2}\rho V^2 S C_L(\alpha) = mg \cdot \frac{\cos(\theta - \alpha)}{\cos\mu} \\ \dot{h} = V \sin(\theta - \alpha) \end{gathered} \tag{4.3.22}$$

Then,

$$\begin{gathered} T(V,N) - \frac{1}{2}\rho V^2 S (C_{D0} + k \cdot C_L^2) = mg \cdot \left(\frac{\dot{h}}{V}\right) \\ \frac{1}{2}\rho V^2 S C_L(\alpha) \cdot \cos\mu = mg \cdot \sqrt{1 - \left(\frac{\dot{h}}{V}\right)^2} \end{gathered} \tag{4.3.23}$$

Then,

$$T(V,N) - \frac{1}{2}\rho V^2 S \cdot \left(C_{D0} + k \cdot \left(\frac{m \cdot g}{\frac{1}{2}\rho V^2 S}\right)^2 \left(\frac{1 - \left(\frac{\dot{h}}{V}\right)^2}{\cos^2\mu}\right)\right) = mg \cdot \left(\frac{\dot{h}}{V}\right) \tag{4.3.24}$$

Rearranging terms, the equation can be written as:

$$\cos^2\mu = \frac{B - B \cdot \left(\frac{\dot{h}}{V}\right)^2}{A - \frac{\dot{h}}{V}} \tag{4.3.25}$$

where A and B are parameters that do not depend on $\dot{h}$ and μ and are constant if the manoeuvre is completed at constant V and N:

$$A = \frac{T(V,N) - \frac{1}{2}\rho V^2 S C_{D0}}{m \cdot g} \qquad B = \frac{m \cdot g \cdot k}{\frac{1}{2}\rho V^2 S}$$

Since these two parameters depend on many other factors that have many sources of error and the value of thrust is not known, experimental data has been used to determine them.

In figure 4.6, the best curve fit can be observed as well as a comparison with the experimental data. The values of A and B found by the least squares method are:

$$A = 0.638 \qquad B = 0.554$$

Then:

$$\cos^2\mu = \frac{0.554\left(1 - \left(\frac{\dot{h}}{V}\right)^2\right)}{0.638 - \frac{\dot{h}}{V}} \tag{4.3.26}$$

In order to find the set of pitch angles that correspond to the vertical velocities calculated above, equations 4.3.19 have been solved assuming the aerodynamic models stated above:

$$\frac{\dot{h}}{V} = sin(\theta - \alpha)$$

$$\frac{1}{2}\rho V^2 S (C_{L0} + C_{L\alpha}\alpha) = mg \cdot \frac{\cos(\theta - \alpha)}{\cos\mu}$$

The results obtained can be observed in figure 4.6.

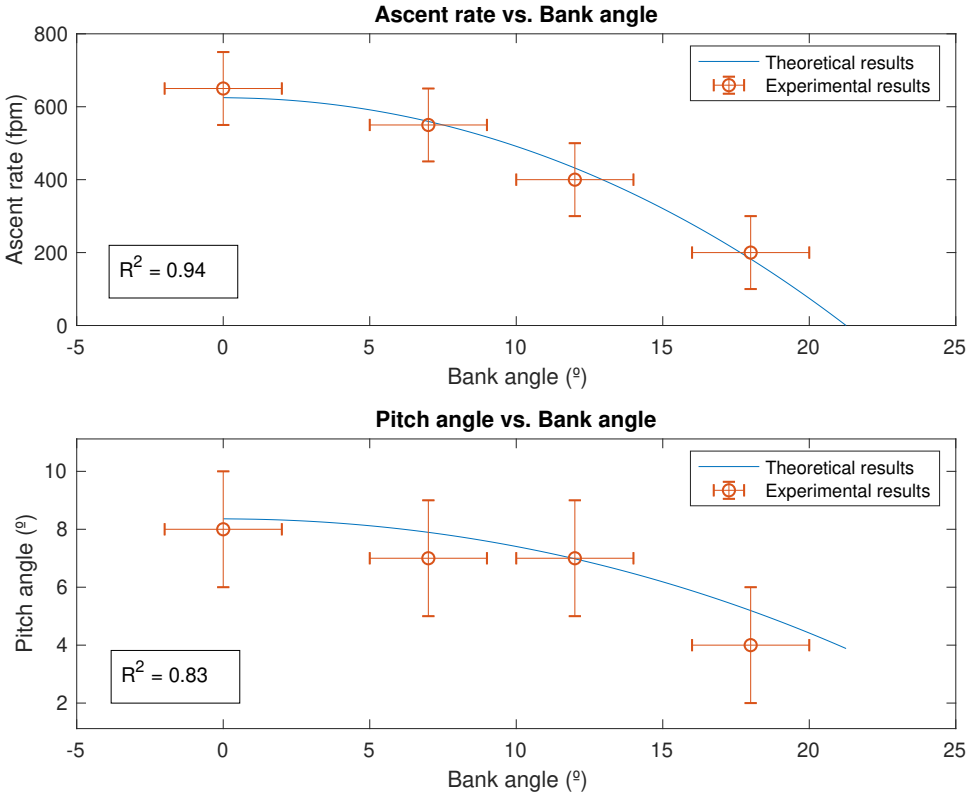


Figure 4.6: Analytic models and experimental results

The analytical solution obtained for both the ascent rate and the pitch angle as a function of the bank angle have a correlation with the experimental results grater than 0.8. Moreover, the model is inside all the error bars of the experimental values. Has to be said, however, that a security factor of 1.2 will be applied to have extra thrust available in case the flight conditions change. The resultant curve that depicts the maximum pairs of pitch and bank angles can be observed in figure 4.7.

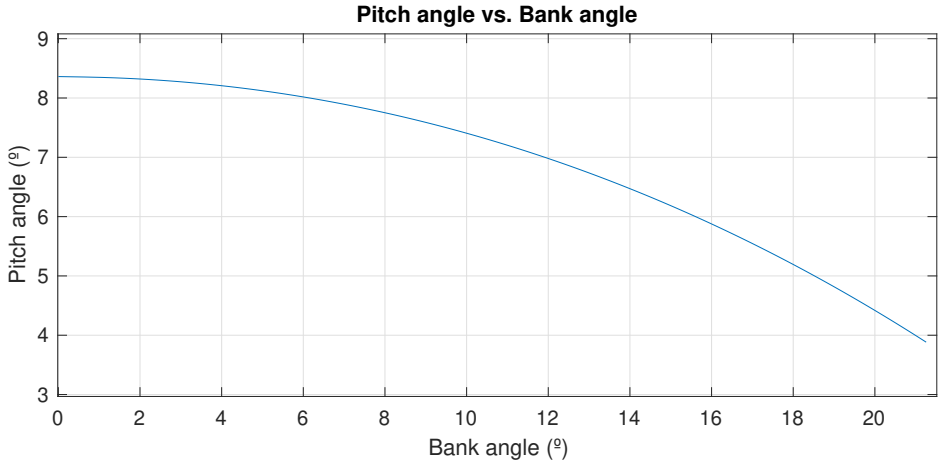


Figure 4.7: Pairs of maximum roll and bank angles with the security factor

Although having defined the relationship between these two parameters, given a situation which the airplane has to correct both: the pitch and bank angle, another condition has to be imposed to chose between all the possible pairs of variables. What it can be done is to force the two corrective manoeuvres (climb and roll) to be executed in such a way that they finish at the same time - which means that ζ and W_z will be reached simultaneously. As it has been explained, the parameters characterizing the aircraft dynamics have been considered stationary during a single manoeuvre. This means that: $\dot{V} = 0$, $\dot{\mu} = 0$, $\dot{\gamma} = 0$. And as a consequence, $\ddot{\chi} = 0$ and $\ddot{h} = 0$. From equation 4.3.19:

$$\int_0^{\zeta} \dot{\chi}\ dt = \int_0^t \frac{g}{V} \tan\mu\ dt = \frac{g}{V} \tan\mu \int_0^t dt \tag{4.3.27}$$

$$\int_0^{W_z} \dot{h}\ dt = \int_0^t V \sin(\theta - \alpha)\ dt = V \sin(\theta - \alpha) \int_0^t dt \tag{4.3.28}$$

Then:

$$\zeta = \frac{g}{V} \tan\mu\ t \tag{4.3.29}$$

$$W_z = V \sin(\theta - \alpha)\ t \tag{4.3.30}$$

If W_z and ζ have to be reached simultaneously:

$$\frac{W_z}{\zeta} = \frac{V^2}{g} \frac{sin(\theta - \alpha)}{\tan\mu} \tag{4.3.31}$$

After computing μ and θ for different $K = W_z\ /\ |\ \zeta\ |$, the following has been obtained:

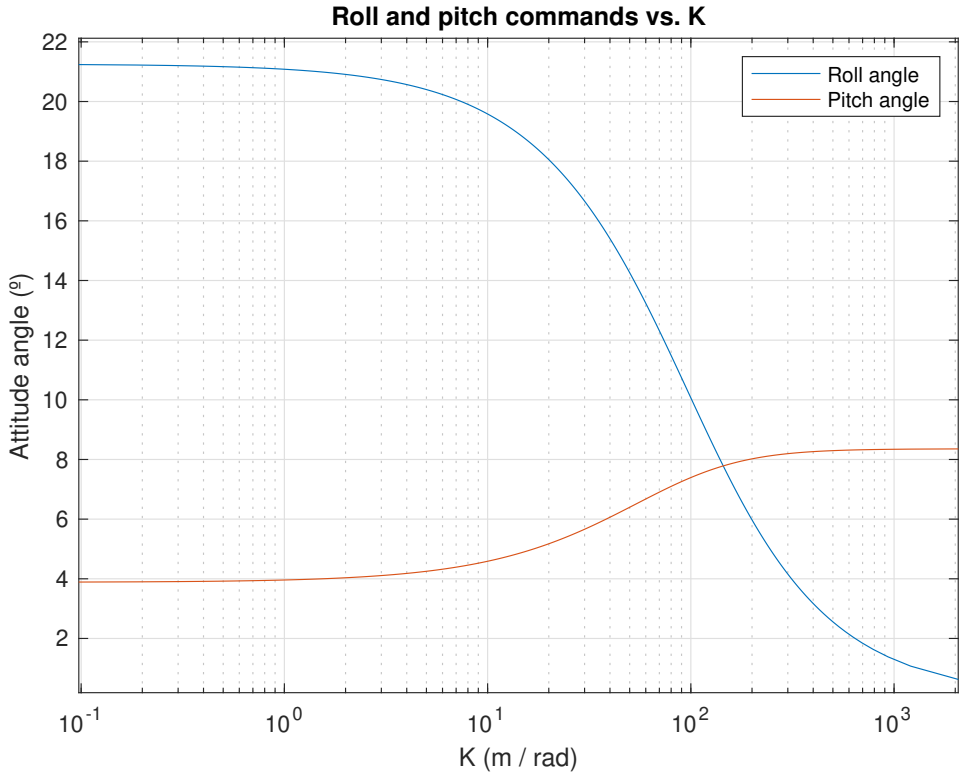
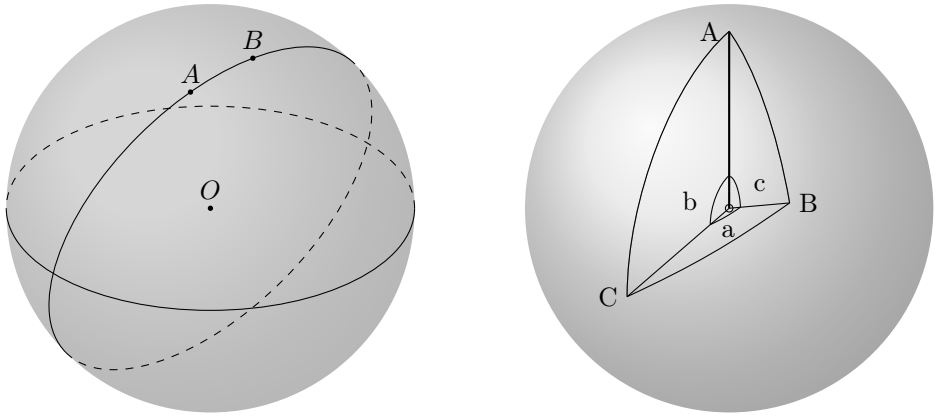


Figure 4.8: Pitch and roll commands for different K

The values obtained from this plot will be used by the flight director to instruct the commands for the different possible values of K.

4.4 3D approach: Navigation analysis

Remembering the problem statement, it has been assumed that the aircraft travels along a Cartesian reference frame where points, represented by x and y coordinates, could be plotted. However, the real aircraft will travel above the Earth and, as a consequence, the angle ζ (error path angle) has to be calculated for a reference frame where points are given in latitude and longitude. To simplify the calculations involved in this problem, the Earth will be considered as a perfect sphere. When doing so, the shortest path between two locations on the globe (i.e. the Earth) can be found as the great circle passing through these two points. A great circle (also known as an orthodrome) is any intersection between the sphere and a plane passing through the center of this sphere [17]. However, as it can be noted, the great circle defining the minimum distance along the globe between two points is the one passing through the two points under analysis, which, together with the center of the sphere define the plane in an unequivocal way. See figure 4.9a.



(a) Great circle joining points A and B. (b) Spherical triangle containing angle $\angle NAW$.

Figure 4.9: Generic spheric traingle with its 6 characteristic angles

If the aircraft maintains a steady path with its wings in a horizontal position, it would fly a great circle around the Earth. Moreover, the track connecting the actual position and the next waypoint is a great circle, too. This means that the angle that has to be calculated is a spherical angle derived from a defined spherical triangle. A spherical triangle is a figure defined in the surface of a sphere limited by three great circles and a spherical angle is one of the three angles formed inside this triangle [17]. A generic spherical triangle is shown in figure 4.9b. As it can be seen, 6 angles can be defined (3 in the surface of the sphere and 3 defined between the segments connecting each node to the center of the sphere). Theoretically, to determine the 6 angles, 3 of them must be known. The equations relating any 3 known angles with the other 3 depend on which angles are known and the relative position between them.

To calculate ζ, it will be suposed that the coordinates of both: the waypoint and the aircraft positions are known. P will be used to depict the aircraft position and B will be the aircraft position at a known previous time, supposing that the aircraft has flown with levelled wings between P and B. Consequently, the aircraft track between P and B is a great circle equivalent to the one that will be flown in the future if the path of the aircraft is not modified. W will be the point representing the next waypoint in the route. The situation is shown in figure 4.10.

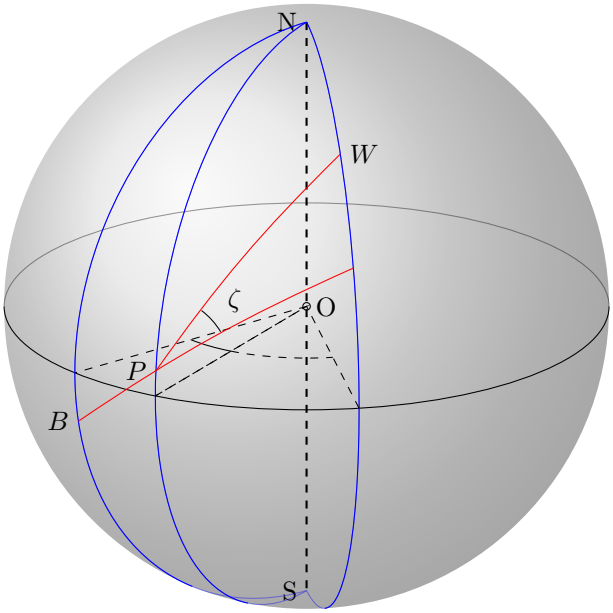
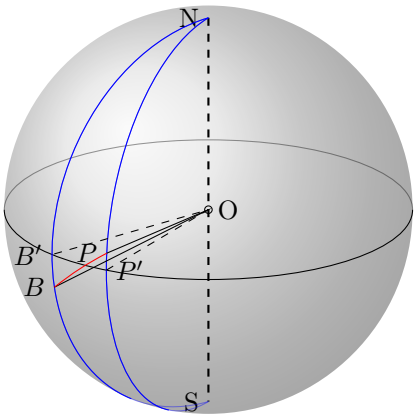


Figure 4.10: Scheme of the generic case where the aircraft path is not aligned to the waypoint direction.

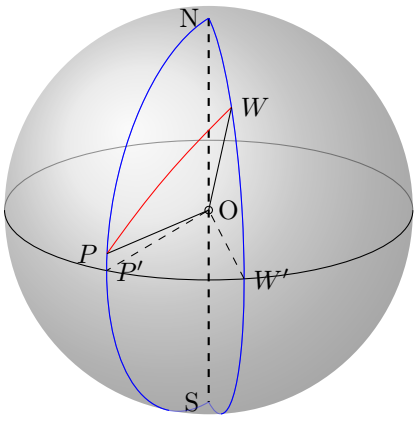
Refering to the figure:

$$\zeta = \angle NBP - \angle NPW$$

Both $\angle NBP$ and $\angle NPW$ are spherical angles of two different spherical triangles. To determine them, the two spherical triangles will be analysed and as it has been explained, in order to do so, 3 angles must be known. See figure 4.11b and 4.11a. Note that (X') refers to the intersection of the parallel passing through point X and the equator.



(a) Spherical triangle containing angle $\angle NBP$.



(b) Spherical triangle containing angle $\angle NPW$.

Figure 4.11: Schemes of the two spherical triangles under analysis.

For the triangle containing the angle $\angle NBP$ and refering to the same nomenclature used in figure 4.9b, the 3 known angles are:

$$P = \angle B'OP' \qquad c = \angle NOP \qquad b = \angle NOB$$

Where, from a navigational point of view and refering to latitude (φ) and longitude (λ):

$$\angle B'OP' = \lambda_P - \lambda_B = \Delta\lambda \qquad \angle NOP = \frac{\pi}{2} - \varphi_P \qquad \angle NOB = \frac{\pi}{2} - \varphi_B$$

For the other triangle, the one containing the angle $\angle WPN$, the 3 known angles are:

$$P = \angle P'OW' \qquad c = \angle NOW \qquad b = \angle NOP$$

And converting the angle to latitude and longitude:

$$\angle P'OW' = \lambda_W - \lambda_P = \Delta\lambda \qquad \angle NOW = \frac{\pi}{2} - \varphi_W \qquad \angle NOP = \frac{\pi}{2} - \varphi_P$$

As it can be noted, the two spherical triangles are of the same type (the same angles are known, although they have different values). This means, that the remaining angles can be found by applying the same equations. Looking at the figures above, one can see that the angle of interest in order to find ζ is the angle C of figure 4.9b. The mathematical relations needed to find it will not be proven; however, as it is well-explained in the equation to find C as a function of A, b and c is:

$$C = \arctan \frac{\sin c \sin A}{\sin b \cos c - \cos b \sin c \cos A}$$

And, substituting for the two spherical triangles:

$$\angle NBP = \arctan \frac{\sin \angle NOP \sin \angle B'OP'}{\sin \angle NOB \cos \angle NOP - \cos \angle NOB \sin \angle NOP \cos \angle B'OP'}$$

$$\angle NPW = \arctan \frac{\sin \angle NOW \sin \angle P'OW'}{\sin \angle NOP \cos \angle NOW - \cos \angle NOP \sin \angle NOW \cos \angle P'OW'}$$

Substituting the angles for latitude and longitude values, the following can be obtained:

$$\angle NBP = \arctan \frac{\cos \varphi_P \sin \Delta\lambda_{PB}}{\cos \varphi_B \sin \varphi_P - \sin \varphi_B \cos \varphi_P \cos \Delta\lambda_{PB}}$$

$$\angle NPW = \arctan \frac{\cos \varphi_W \sin \Delta\lambda_{WP}}{\cos \varphi_P \sin \varphi_W - \sin \varphi_P \cos \varphi_W \cos \Delta\lambda_{WP}}$$

4.5 Automation analysis: control action

The control action required is a relationship between the current and previous locations and the following waypoint position and the pitch and roll attitudes to arrive at the mentioned waypoint in an efficient way [18]. Many algorithms can be used to perform such calculations: in this section, three different proposals that could be used to solve the current problem will be explained. Then, a single one will be selected to develop the flight director prototype.

As it has been seen, the pair pilot-airplane can be modeled as a non-linear MIMO plant (multiple input, multiple output), then the methods applied to control such system have to take the characteristics of this complex architecture into account. The controllers that would be discussed will be: a pair of PID controllers with dynamic saturation, a linear-quadratic regulator and a model predictive controller [19].

4.5.1 Coupled PID controllers

One of the most simple and robust control methods for SISO systems is the use of a PID controller. which minimizes the error by applying the following correction through its actuator:

$$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t) + K_I \int_{-\infty}^{t} (x_{des} - x_t)dt \tag{4.5.1}$$

However, in the current problem two PID controllers are necessary to control the MIMO system. Moreover, and since the outputs are limited to predefined values (a maximum and minimum pitch and roll are allowed), the two PID outputs have to be saturated. Consequently, the architecture of the controller would be two independent PID with dependable saturations (see figure 4.12).

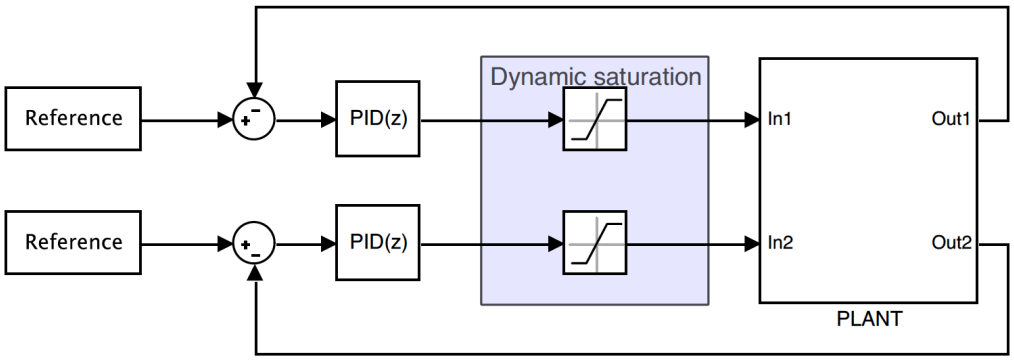


Figure 4.12: Scheme of two coupled PID with dependable saturation

4.5.2 Linear-quadratic regulator

A LQR controller consists of an algorithm used to operate dynamic systems at minimum cost. Let's suppose that a linearised system can be described in the following way [20]:

$$x_{k+1} = A \cdot x_k + B \cdot u_k \tag{4.5.2}$$

Where x_k represent the state-space variables at iteration k and u_k represent the actuator states at the same iteration. The objective of the control is to drive the initial state to the smallest possible value in a predefined time interval. Compared to a conventional controller, the cost function takes into account u_k which means that the minimization of the actuators effort is also considered. The cost function is described below:

$$J = \sum_{k=0}^{\infty} \left(x_k^T \cdot Q \cdot x_k + u_k^T \cdot R \cdot u_k\right) \tag{4.5.3}$$

Where matrices Q and R represent the weights that the designer wants to give to each state-variable to be minimized. One of the limitations of the LQR is that the linearisation of matrices A and B is done by taking a single equilibrium point so that the model is time-invariant. When the system is not in the equilibrium point (for example a turn), the controller is not efficient and, in some cases, it can lose robustness. A possible improvement to this method is the continuous

linearisation of matrices A and B evaluated to the point where the plant is found in each moment. By doing so, a better efficiency would be achieved.

4.5.3 Model predictive control

A MPC is an advanced method to control MIMO systems. The idea behind the algorithm is to solve the problem using optimization methods while satisfying a set of constraints that can involve state-space variables as well as actuators. This method is highly useful when considering systems as the one proposed: the operational range of the different space-state variables could be defined and the regulator would make the pilot fly in the most optimal way. For example the system output could take into account limitations of: thrust, angle of attack, pitch, roll and velocity.

4.5.4 Development of the final solution

The selected controller to develop the flight director given its robustness and simplicity of implementation will be the two PID controllers with dependable saturation. The saturation of the outputs will depend on the ratio between the two errors (difference in altitude and track angle), which has been defined as K. The outputs of roll and pitch angle will be the ones shown in figure 4.7. Doing so, the flight will be divided in two different states: the first one will be characterized by a $W_z \neq 0$ and/or $|\ \zeta\ | \neq 0$ and the second one will have $W_z = |\ \zeta\ | = 0$. When the system identifies a situation defined by the first case, a corrective manoeuvre will be needed. As it has been explained, for design purposes and trying to simulate as much as possible the flight manoeuvres when flying in general aviation, it has been decided to correct both: the angle and altitude deviation simultaneously. In this way, the manoeuvre will be developed in constant bank and pitch angles until the outputs of the PID controllers are smaller than the ones defined as a function of K. Then, the PID will take charge of displaying the bank and pitch attitudes to assure a smooth transition between both states. The general block diagram proposed is shown in figure 4.13.

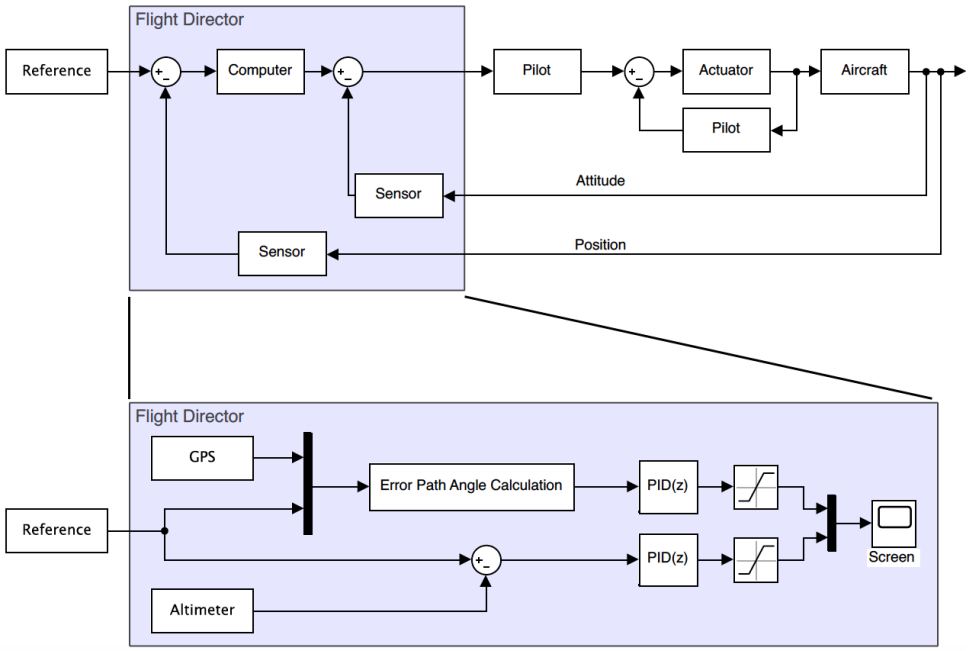


Figure 4.13: Control action block diagram

The flight director has three different inputs (aircraft position, waypoint position and attitude). To simulate the control action and tune the PID parameters involved in the problem, a Simulink program has been developed (see Annex A). Four main blocks have been created each one simulating an important actor of the entire system (Pilot, Aircraft dynamics, Kinematic model, Control action). The set of equations that sum up this modelization are shown below:

Pilot: The pilot has been modelized as a transfer function of gain K = 1, with a non-constant delay τ that follows a normal distribution with mean $\mu = 0.6s$ and $\sigma^2 = 0.1s^2$. More specifically, this transfer function represents the interpretation of the flight director display by the pilot and the consequent action on the aircraft to reach a specified attitude [21]. This values have been obtained from the results of a experiment simulating time delay of pilots in a Cessna 152 [22]. Consequently:

$$P(s) = e^{-s\tau} \qquad where \qquad \tau \sim N(0.6,\ 0.1)$$

Aircraft: The modelization of the aircraft dynamics has been previously explained. The equations that have been used are:

$$\begin{aligned} T(V,N) - \frac{1}{2}\rho V^2 S C_D(\alpha) &= mg \cdot \sin(\theta - \alpha) \\ \frac{1}{2}\rho V^2 S C_L(\alpha) &= mg \cdot \frac{\cos(\theta - \alpha)}{\cos\mu} \\ \dot{\chi} &= \frac{g}{V}\tan\mu \\ \dot{h} = -\dot{z_h} &= V\sin(\theta - \alpha) + V_{az} \end{aligned} \tag{4.5.4}$$

The aerodynamic curves of the aircraft have been obtained from: [16] And have been modelled as:

$$C_L(\alpha) = 4.77\alpha \ + \ 0.35$$
$$C_D(C_L) = 0.07C_L^2 \ + \ 0.027$$

Kinematic model: Regarding the kinematic model, the equations have been derived from the hypothesis of a spherical Earth. Moreover, wind perturbations have been added to the model in order to validate it in such conditions:

$$\dot{\varphi} = \frac{V\cos\chi - V_a cos\chi_w}{R_e} \tag{4.5.5}$$

$$\dot{\lambda} = \frac{V\sin\chi + V_a cos\chi_w}{R_e\cos(\varphi)} \tag{4.5.6}$$

where:
V_a is the wind velocity
χ_w is the wind velocity angle
R_e is the Earth radius.

Control action: As previously stated, the control action will consists of two independent PID controllers which outputs will be saturated as a function of the magnitude of the attitude and track angle deviations.

First PID: Horizontal Control
The error that will be minimized is obtained from the navigational problem resolved above:

$$\zeta = \angle NBP - \angle NPW$$

Using a reference of $\zeta = 0$, the controller will constantly minimize the track deviation. The type of controller used will be a parallel PID in discrete time, which can be expressed as a transfer function:

$$L(z) = K_p + K_i \cdot T_s \frac{1}{z-1} + K_d \frac{N}{1 + N \cdot T_s \frac{1}{z-1}} \tag{4.5.7}$$

The outputs will be then saturated between: $[-\mu_{MAX}(K), \mu_{MAX}(K)]$. The values for the tuning of the PID controller will be calculated using the developed model in Simulink.

Second PID: Vertical Control

The error that will be used for this second PID will be the difference between the current and desired altitude, which can be expressed in the following way:

$$\epsilon = Alt_W - Alt_P = W_P \tag{4.5.8}$$

The controller will be designed in the same way as the one above:

$$L(s) = K_{pA} + K_{iA} \cdot T_s \frac{1}{z-1} + K_{dA} \frac{N}{1 + N \cdot T_s \frac{1}{z-1}}$$

The outputs will be then saturated between: $[\theta_{MIN}, \theta_{MAX}(K)]$.

The tuning method has been done using an heuristic try-and-error approach using reference values from other known flight director commands. Since the flight conditions can vary between different conditions of aerodynamic velocity, wind speed, upward and downward air current between others, the tuning process has been done for a set of standard conditions. Once the value for the parameters have been decided, some perturbations have been added to the model in order to validate the controller for changing velocities, wind and vertical currents of air.

In order to tune the PID controllers and run the simulations, a specific aerodynamic velocity range has to be decided. To simplify the calculations and simulate the reality as much as possible the following has been supposed:

$$V = \begin{cases} 70\ kts & \text{if } |\ \zeta\ | > 0 \text{ and } W_z > 0 \\ 90\ kts & \text{in any other case} \end{cases} \tag{4.5.9}$$

The results obtained have been:

	Horizontal PID	**Vertical PID (Sub-index A)**
K_p	0.9	0.004
K_i	0.0	0.0003
K_d	0.0	-0.008 (N = 0.1)

Table 4.3: PID parameters for vertical and horizontal control.

As it can be seen in Table 4.3 or the horizontal flight path controller, only a proportional controller has been needed: this is due to the fact that the parameter controlled has a value of zero for the equilibrium position (no bank angle is needed for a straight and levelled flight). The architecture of the system integrates the velocity to obtain position and the integrator used eliminates the error in the same way as an integrator of a PID would do. However, the vertical PID controls the pitch variable (which is not zero) for the equilibrium position: consequently, the error has to be integrated with a PID. A derivative controller has also been added to smooth the transition between climb and levelled flight.

The algorithm that represents the control action is:

```
while not above last waypoint do
    if above next waypoint then
        Change next waypoint;
    else
        Calculate track angle error;
        Calculate altitude error;
        Desired Pitch ← Pitch from pitch action PID;
        Desired Roll ← Pitch from pitch action PID;
        if Altitude error > 0 then
            Calculate K;
            if Desired Pitch > Maximum Pitch (K) then
                Desired Pitch ← Pitch from K calculation;
            end
            if | Desired Roll | > Maximum Roll (K) then
                if Desired Roll > 0 then
                    Desired Roll ← Roll from K calculation;
                else
                    Desired Roll ← - Roll from K calculation;
                end
            end
        else
            if Desired Pitch < Predefined Minimum Pitch then
                Desired Pitch ← Predefined Minimum Pitch;
            end
            if | Desired Roll | > Maximum Roll (K = 0) then
                if Desired Roll > 0 then
                    Desired Roll ← Maximum Roll (K = 0);
                else
                    Desired Roll ← - Maximum Roll (K = 0);
                end
            end
        end
        Display Desired Roll - Current Roll ;
        Display Desired Pitch - Current Pitch ;
    end
end
```

To validate the model, a simulation has been done with no wind conditions and three waypoints at different altitudes: Sant Sadurní d'Anoia (41.42° N, 1.79° E at 3000 ft AMSL), Vallbona d'Anoia (41.52° N, 1.70° E at 4500 ft AMSL) and Sant Esteve Sesrovires (41.50° N, 1.85° E at 3500 ft AMSL). With an intital heading of 000° at 3000 ft AMSL above Martorell (41.48° N, 1.92° E). The results obtained are shown in figure 4.14.

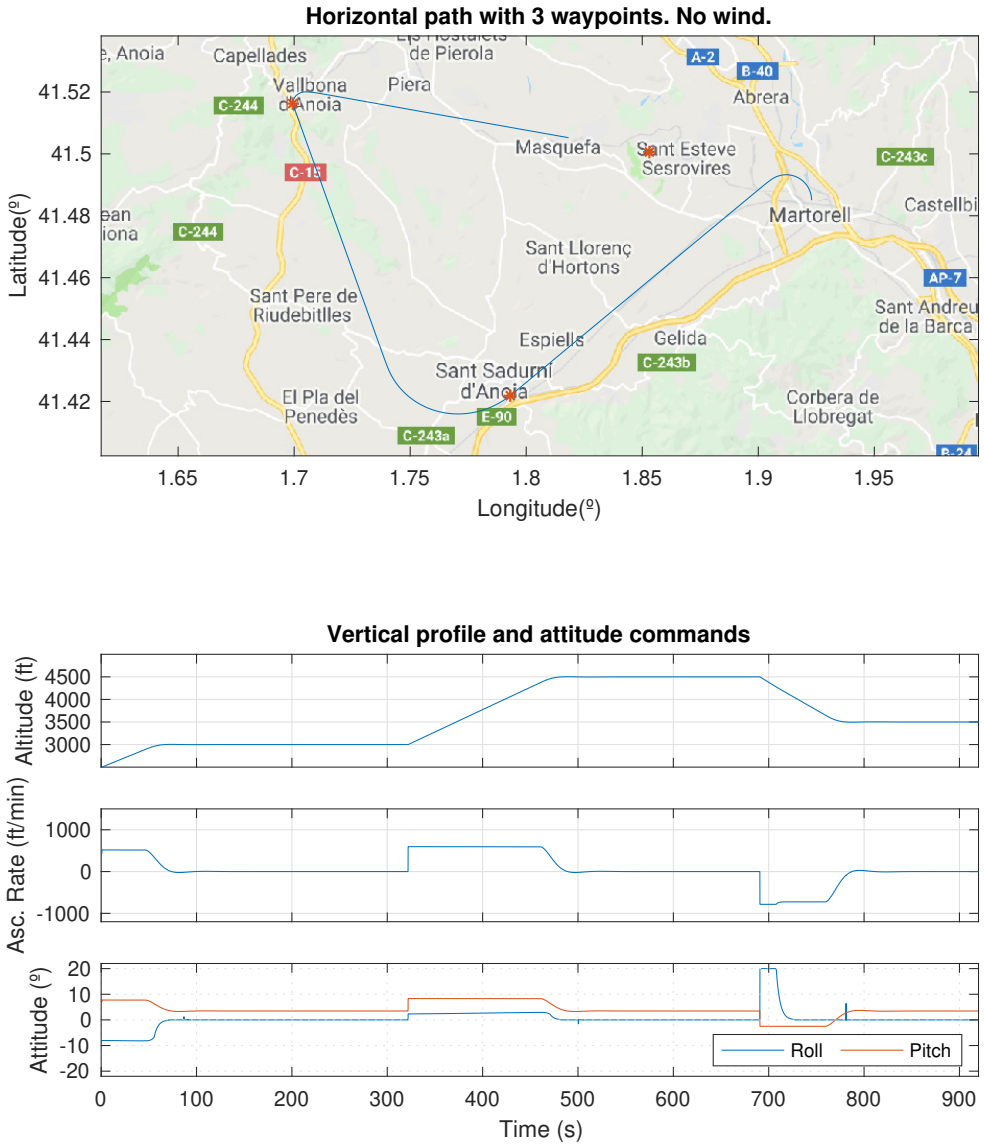


Figure 4.14: Simulation of the aircraft horizontal path, vertical profile and attitude commands for 3 consecutive waypoints.

The flight starts at an altitude of 3000 ft, which coincides with the altitude of the first waypoint: consequently, the first corrective manoeuvre is completed with full bank angle and almost no change in pitch angle. After this, the airplane continues the flight until St. Sadurní d'Anoia where it does the second change of attitude: since the second waypoint is at a higher altitude than the first one, both the bank and pitch angles are limited by the calculation of the K parameter and then the turn radius is bigger than the first one. After 80 s, the airplane levels wings and recovers pitch attitude. The third waypoint is at a lower altitude than the second one, consequently, both roll and pitch angles are not limited as a function of K and a roll of 20° and a pitch angle of -2.5° are commanded. Regarding the ascent rates, the values are very similar to the ones obtained experimentally.

Although the control action is stable for the design conditions (known velocity, no wind...), the model has to be tested for different conditions such as strong wind from different directions, a range of aerodynamic velocities and different initial conditions.

Horizontal Path for different initial headings. In the figure below, the simulation of the aircraft path can be observed for initial conditions of 000°, 090°, 180° and 270°; which represent different initial error track angles. In all cases, the control action employed is stable.

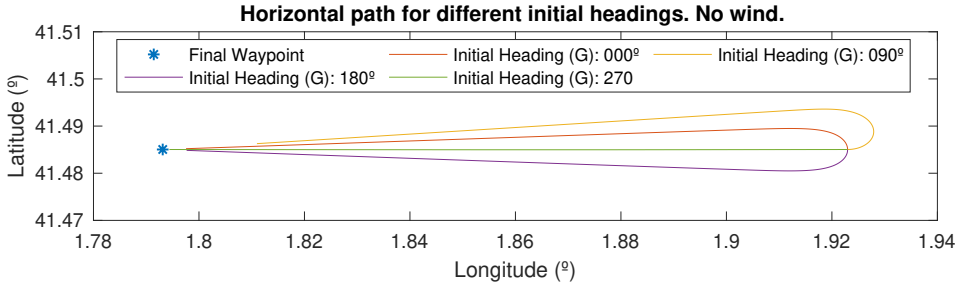


Figure 4.15: Simulation of the aircraft horizontal path for different headings between hypothetical waypoints

Horizontal Path for different wind conditions. The other perturbation tested has been the incorporation of strong winds. In the next figure, four flights with wind conditions, of 40 kts each one, from different directions have been compared to a standard flight with no wind.

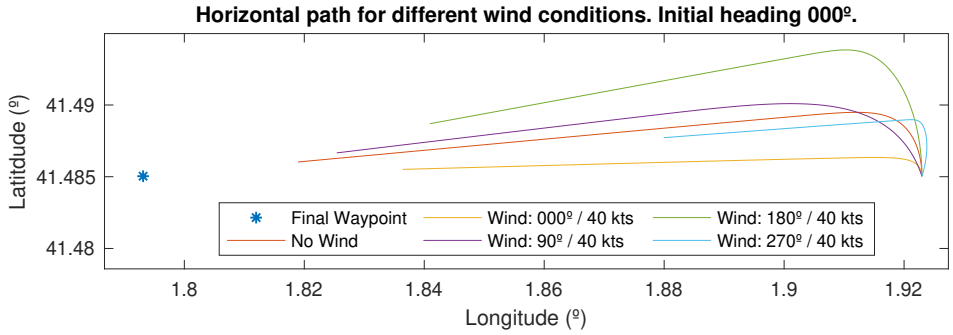


Figure 4.16: Simulation of the aircraft horizontal path for different wind conditions

Horizontal Path for different aerodynamic velocities. The last perturbation tested has been the modification of the aerodynamic velocity. A range of velocities from 60 kts to 105 kts has been simulated in the figure below.

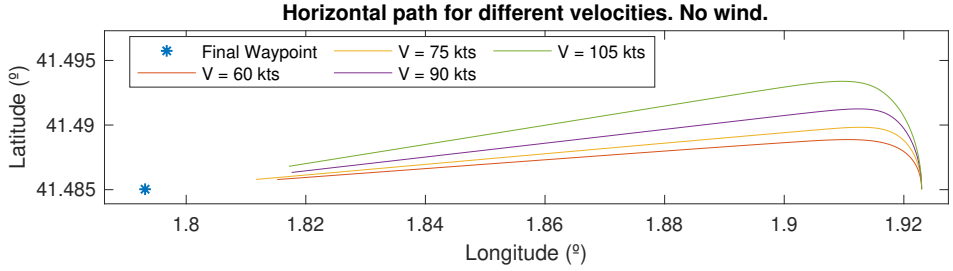


Figure 4.17: Simulation of the aircraft horizontal path for different aerodynamic velocities

Vertical profile for different upward/downward currents of air. The control action behaviour has been analysed when experimenting vertical air streams with an aircraft air velocity of 70 kts. The model is stable for a range of $[-100, 100]$ fpm air currents (see figure 4.19) although an increment on the downward air stream produces a decrement on the ascent rate.

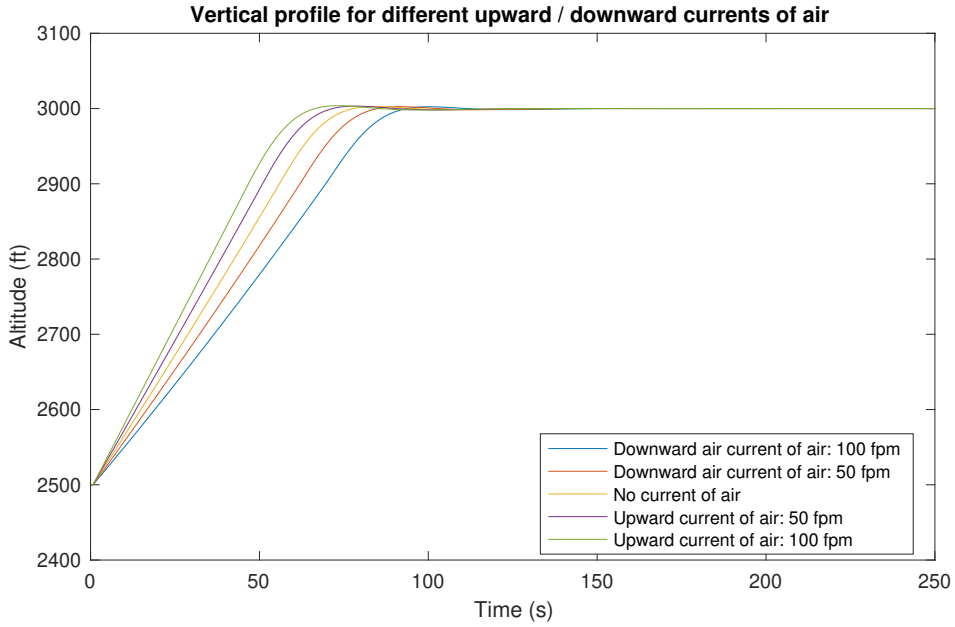


Figure 4.18: Simulation of the aircraft vertical path for different vertical currents of air

Vertical profile for different airspeed velocities. The vertical profile has also been tested for different airspeed velocities since it is difficult to maintain an exact airspeed of, exactly, 70 kts. As it can be seen in figure 4.19, the system is stable for the entire range of velocities analysed.

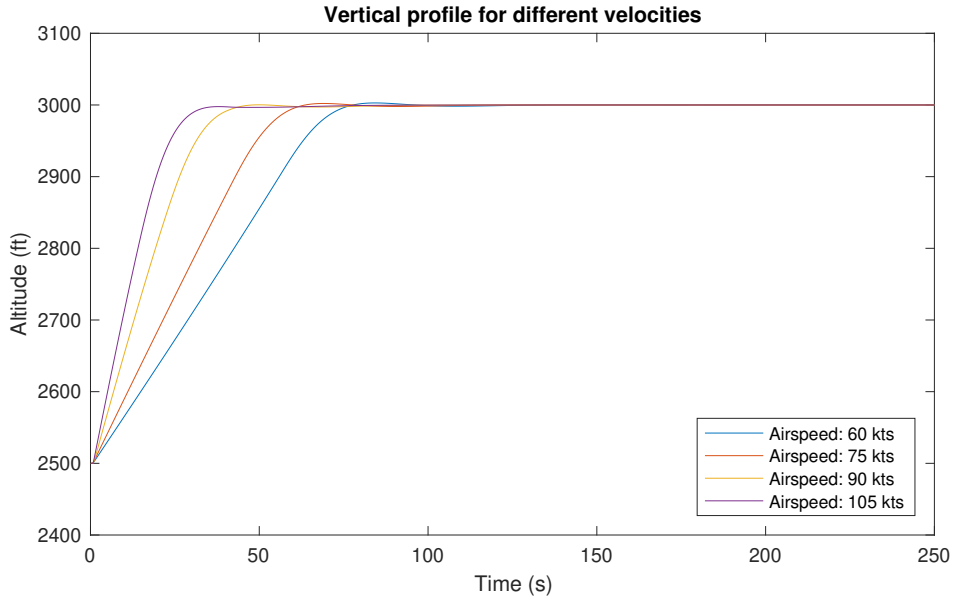


Figure 4.19: Simulation of the aircraft vertical path for different velocities

4.6 iOS application development

Swift programming language has been used to develop the application containing the flight director. This programming language has been developed by Apple Inc. and it is designed to work with Cocoa, Objective-C, C and C++. In this section two main points will be discussed: on the first hand, the sensors used to develop the application will be defined and the data quality will be analysed. On the second hand, the graphic interface design process will be explained.

4.6.1 Sensors used and data treatment

One of the initial requirements to develop the flight director was its total independence from the aircraft systems. Demand that has been taken into account when designing the algorithm and choosing the necessary inputs to compute the command outputs. Two kind of inputs can be distinguished: direct (or non-processed) such as: position, altitude, roll and pitch and indirect (or processed) such as: course. The aircraft horizontal position is necessary to calculate, together with the course, angle ζ. And the current altitude to calculate the altitude deviation W_z. Then, the aircraft course is necessary to complete the calculus of ζ. The measurement of the direct inputs could be done through one single sensor or the fusion of many. And the calculation of the indirect measurements has to be done by, mathematically, processing the direct measurements. The hardware used (the iOS platform) has the following sensors that can be used to calculate the inputs needed: GNSS receiver, accelerometer, magnetometer and barometer. The precision of the localization data received can be adjusted using the software by modifying the energy used by the application. But, in general terms using a specific instrument, the GNSS precision depends on the quantity of satellites tracked and its relative position (factor independent of the user). To improve it, fusion algorithms can be used to combine, for example, the GNSS position with the accelerometers included in the iPhone. Kalman filter can be an option when considering such methods.

The algorithm consists of a recursive method that estimates the position of the subject using the combination of two or more measures (in the current case: GPS and accelerometers). The method used needs to know the variances of both measurements in order to process the result [23]. It is for this reason that, if the hardware comes with an integrated software with a Kalman filter implemented, it is worth using it: the calculation of the variance of the measurements would be integrated on the algorithm and the efficiency of the method would be maximized.

The Swift programming language comes with a service called *Core Location* [24] that provides a precise geographic location, altitude and orientation using all the necessary sensors included in the device such as: Wi-Fi, GPS, Bluetooth, magnetometer and barometer. Since, the object also calculates geographical course, it will be used to calculate all the direct and indirect inputs for the flight director.

To have an idea of the quality of the CoreLocation data (in terms of filtering capacities and smoothness of the final result), longitude, latitude, course and altitude AMSL have been registered for a period of a flight of 90 seconds using the mentioned Swift object. The flight consisted of two consecutive ascent turns from heading 240° to heading 160° and back to heading 200° from 1700 ft to 2200 ft. The registered data can be observed in Figure 4.20. Moreover, a comparison between the course calculated manually (with spheric trigonometric equations) and the course calculated by the CoreLocation object is displayed.

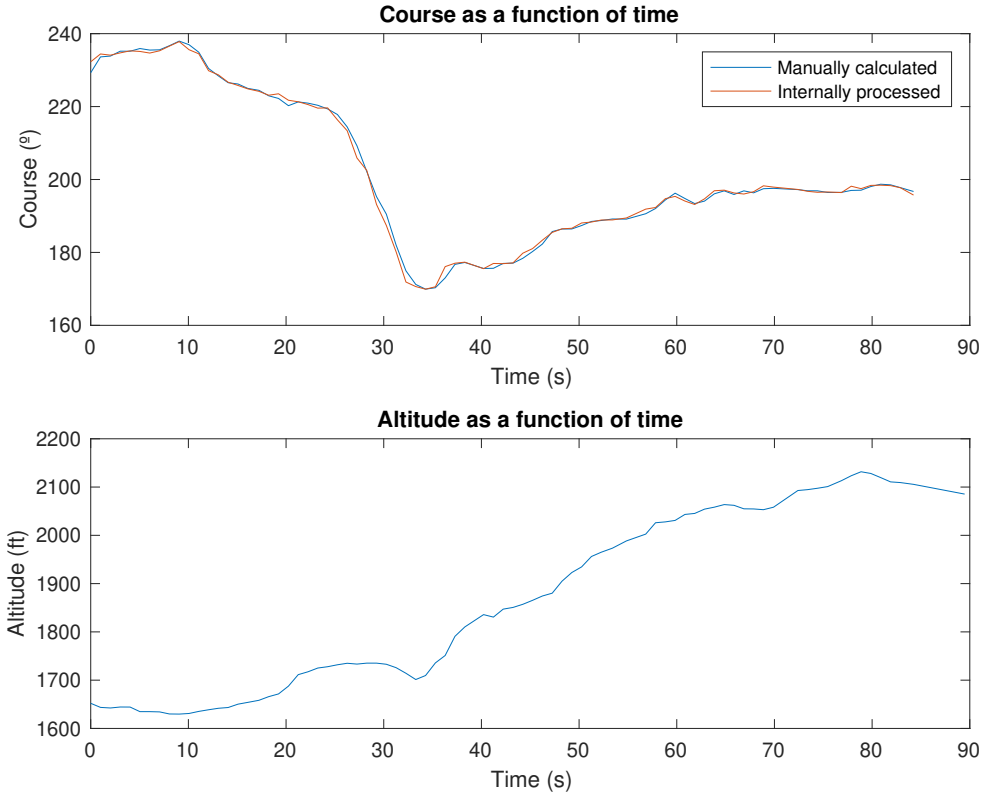


Figure 4.20: Experimental data of course and altitude

As it can be seen the data displayed is continuous and both the altitude and course values are correct in general trends. On the other hand, the course computed by the Core Location object is very similar to the one computed manually using the equations explained previously. From now on, the geographical data filtered by the internal algorithms of the Swift language will be considered correct and both the direct and indirect calculations done by the Core Location object will be used to compute the command indications of the flight director.

4.6.2 Graphic interface

The iOS application graphic interface has been developed using Swift predefined objects (such as buttons) and a class named *UIBezierPath* included in the iOS development tool. This class consists on simple instructions that define simple shapes or paths with specific attributes such as its size, colour, etc. A single Bézier path object can contain many subpaths that have the same attributes as the main one and each object is defined by a series of sequential points. Two main views have been programmed: the first one, consisting on a caution message to prevent the user from using the application as a 100% reliable source for piloting and navigating (see figure 4.21); and the second one consisting on the flight director superimposed on a artificial horizon (see figure 4.22). The application will be tested on an iPhone XR.

Figure 4.21: First view of the application developed

Regarding the graphical interface that has the artificial horizon with the command bars on it, the design has been based on similar instruments such the artificial horizon of the GARMIN G1000 or that of the Airbus family. The artificial horizon has, then, two main parts: a blue surface representing the sky and a brown one representing the ground. The line dividing the two areas varies with pitch and roll to represent the attitude of the aircraft. Additionally, the instrument has a black diagram with a center dot and a wing at each side that represent the airplane. When flying, the aircraft diagram is static with respect to the screen and the movable parts are the blue and brown areas. This is due to the fact that, since in a turn or a pitch movement the instrument screen follows the aircraft motion, the pilot has the illusion that it is the airplane diagram what is moving and the artificial horizon what remains static. Moreover, the instrument contains white lines that help the pilot with the piloting of the aircraft. In first place, an arc has been placed to indicate the roll of the aircraft: it shows indications from -60° to 60° with small lines indicating each 10° of bank angle and big lines showing multiples of 30°. On the other hand, horizontal lines have been placed above the artificial horizon to display the pitch attitude of the airplane. Small bars indicate increments of 5° from the horizon and 10° are displayed with bigger bars. Not all the bars are always showed but only those that are contained in an interval of $[-35°, +20°]$ with respect to the current pitch.

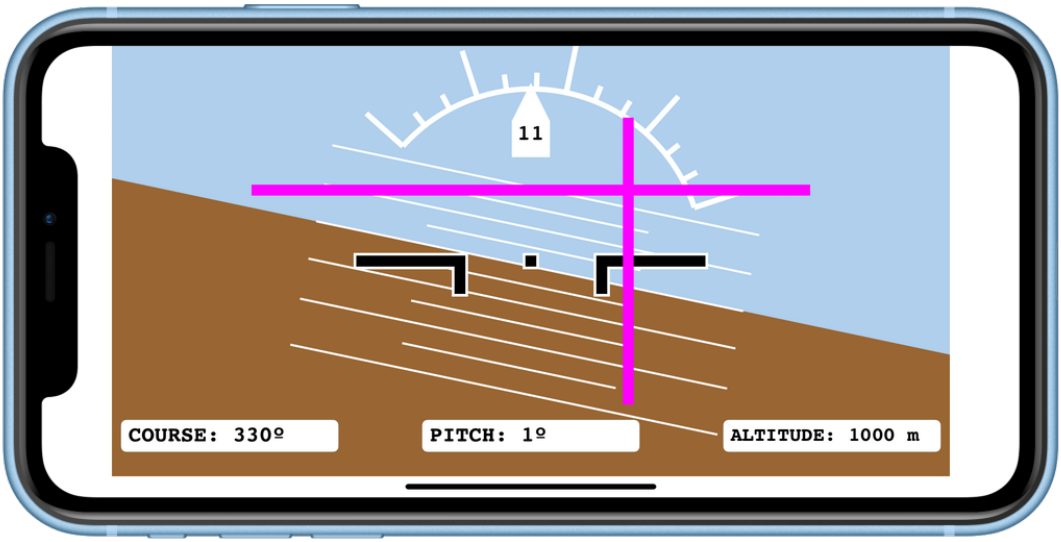


Figure 4.22: Second view of the application developed

The design of the movable parts of the artificial horizon (blue, brown and white figures) has been done by programming them on a static reference frame and moving it with two posterior rotations (pitch and roll). (See figure 4.23).

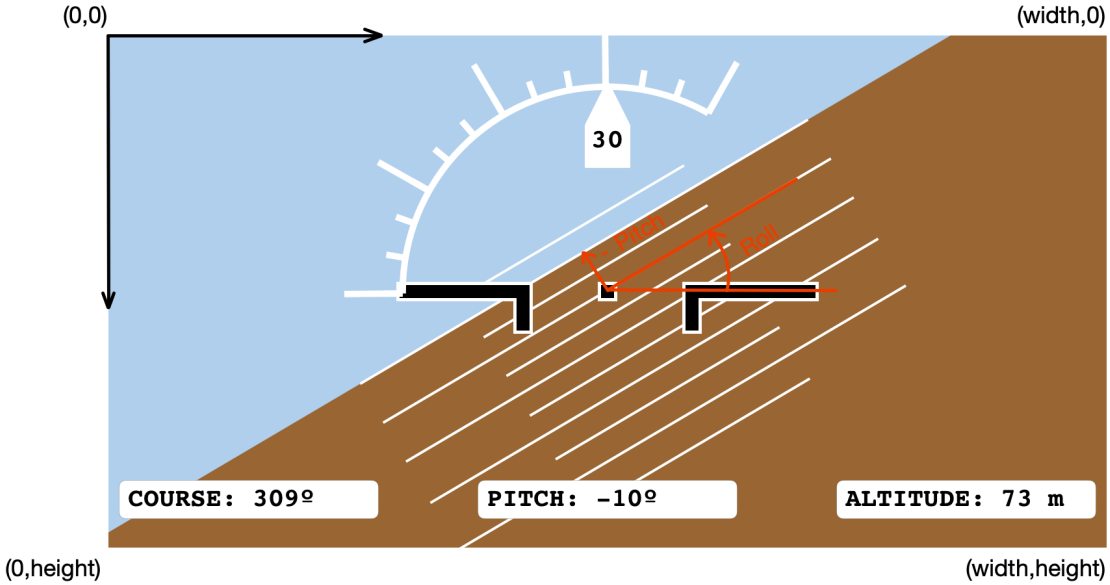


Figure 4.23: Design of the movable parts of the artificial horizon

The two consecutive rotations have been made with respect to the center of the screen by first rotating around the pitch axis and then rotating it around the roll axis. If we refer a generic point

(x_1, y_1) that wants to be transformed to a point coupled with the movable part of the artificial horizon (see figure 4.23), the following corrections have to be done:

Pitch rotation The pitch correction will consists on maintaining the same x coordinate and decreasing the y coordinate as the pitch increases to simulate a rising terrain. Then, if the coordinates of the new point are (x_2, y_2). The transformation is as follows:

$$\begin{aligned} x_2 &= x_1 \\ y_2 &= y_1 \cdot \tan(-\theta) \end{aligned} \tag{4.6.1}$$

Roll rotation The roll correction, however, consists on a simple rotation about a normal vector to the screen. The angle rotated is: $-\mu$. As stated above, this is done to simulate a rotating terrain when the aircraft rotates in the opposite direction.

$$\begin{aligned} x_3 &= x_2 \cdot \cos(-\mu) - y_2 \cdot \sin(-\mu) \\ y_3 &= x_2 \cdot \sin(-\mu) + y_2 \cdot \cos(-\mu) \end{aligned} \tag{4.6.2}$$

As it can be seen in figure 4.23, the reference frame origin is located on the most upper-left corner. Consequently, each rotated point would be located at a position:

$$\begin{aligned} x &= \frac{\textit{screen width}}{2} + (x_1 \cdot \cos(-\mu) - y_1 \cdot \tan(-\theta) \cdot \sin(-\mu)) \\ y &= \frac{\textit{screen height}}{2} - (x_1 \cdot \sin(-\mu) + y_1 \cdot \tan(-\theta) \cdot \cos(-\mu)) \end{aligned} \tag{4.6.3}$$

Moreover, to give more precise indications, 4 numerical values are given to the user: Bank angle, course (referenced to the true north), pitch and altitude (in metres AMSL).

To built the flight director, two magenta command bars have been added. The maximum displacement of both lines is 1/6·Height and 5/6·Height for the vertical one and 1/6·Width and 5/6·Width for the horizontal one: which correspond to indications within a range of $\mu \in [-22°, 22°]$ and $\theta \in [-5°, 12°]$. It has to be remembered, that the information displayed by the bars is not the desired attitude, but the correction to be applied to the current one, to obtain the desired pitch and roll.

4.6.3 Application algorithm

The algorithm that runs the application is the one described in the control action explained above (see Section 4.5) translated to Swift language. The time increments used to refresh the information to compute the position of the magenta command bars is $T_s = 0.1s$; which has been considered sufficient for the method used. Regarding the PID controller, the derivative calculation is done by simply applying the Newton finite difference formulation and the integral calculation by applying the rectangle rule. Then, in the final application algorithm, the PID reads as follows:

$$\begin{gathered} U = k_p \cdot \epsilon_k + k_i \cdot I_k + k_d \cdot N \cdot \frac{\epsilon_k - \epsilon_{k-1}}{T_s} \\ \text{where:} \\ I_k = I_{k-1} + \epsilon_k \cdot T_s \end{gathered} \tag{4.6.4}$$

To computation of the pitch command is the result of the saturation of a PID controller (contrary to the roll signal which only consists on a saturated P controller). This formulation can produce erroneous outputs because of the integral windup. The integral windup occurs when the

output signal reaches its limit and the integral controller does not affect the controlled system. In this case, the integral term continues accumulating and increasing. The problem comes when the controller returns to the effective zone: because of the high value of the integral term, the output continues rising resulting in a high response time and an overshooted signal [25]. To solve this, what it has been done in the flight director algorithm is to reset the integral term once the controller enters the saturated zone. By doing this, the integral controller only accumulates error if it is inside the effective zone.

Another particularity of the application algorithm is that it is needed to find a way to determine when a waypoint is crossed in order to change the target point coordinates to recalculate the error and follow a new direction. To do so, the distance between the current aircraft position and the next waypoint will be calculated and if the result is smaller than a predefined value (1000 m will be used for the first iteration), the following waypoint will be selected. To calculate the distance between two points above a sphere, the great circle sector length between the two mentioned points has to be computed. To obtain a precise result, the spherical law of cosines is used [26]:

$$\Delta\sigma = \arccos\big(\sin\varphi_1 \sin\varphi_2 + \cos\phi_1 \cos\varphi_2 \cos(\Delta\lambda)\big) \tag{4.6.5}$$

And the distance is then simply computed by:

$$d = r \cdot \Delta\sigma \tag{4.6.6}$$

This formulation, however, presents some limitations when the points are close one from the other because of the low floating-point precision of the software used and the consequent inexactitude when computing the cosine of an angle. It is for this reason that ($\Delta\sigma$ will be calculated using the haversine formula [27]:

$$\begin{aligned} \Delta\sigma &= \operatorname{archav}\left(\operatorname{hav}\left(\Delta\varphi\right) + \cos\varphi_1 \cos\varphi_2 \operatorname{hav}\left(\Delta\lambda\right)\right) = \\ &= 2\arcsin\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)} \end{aligned} \tag{4.6.7}$$

5 | Results and Validation: Flight tests

In order to validate the results of the algorithm, a series of flight tests have been conducted. Generally, the tests consisted on a local flight (from LELL to LELL) on a Cessna 152 with two people on board. During a period of the flight, one of these two people followed the commands of the flight director installed in the airplane (see Figure 5.1) while the other person was constantly monitoring the state of the flight for security reasons. The flight test was considered initiated once the airplane was outside the LELL ATZ and finished when the airplane crossed the last waypoint on the route. Since the take-off field was a controlled aerodrome, the traffics were requested to follow standard departures and the flight director commands could not be trusted for ATC compliance reasons. The data taken to analyse the results were GPS coordinates (latitude and longitude each 10 seconds) and altitude AMSL. The tracker was a GPS logger incorporated in the aircraft, totally independent from the iPhone that showed the flight commands. Before initializing the flight, the flight director was checked to identify misalignments or faulty results. Before starting the engine, the iPhone was put in a correct position so that the application showed an indication of 0° of pitch and roll.

Figure 5.1: Photography of the flight director installed in the airplane

The security measures taken during the flight tests consisted on making the co-pilot monitor certain parameters periodically. These controls included the supervision of the airspeed (to avoid stalls, spins and structural damage of the airplane); a secure engine RPM operational range (to avoid ice formation in the carburettor or a dangerous excess of the engine speed) and dangerous aircraft attitudes (of pitch and roll). To do so, a check list was written and used, during the flight tests, by the pilot (see Table 5.1).

Test Flight Check list ***Cessna 152***		
Airspeed Indicator	Above 55 kts Below 100 kts	GREEN ARC
RPM	Above 2000 Below 2500	GREEN ARC
Pitch (Vertical indication)	Above -5° Below 10°	-
Roll (Circular indication)	Not excess 30°	-

Table 5.1: In-flight check list for co-pilot.

Moreover, the co-pilot was in charge of controlling the airplane position at every moment to avoid entering the LELL ATZ without previous notification (since it is a Class D airspace from 500 ft AGL and 3500 ft AMSL) or entering the LEBL controlled airspaces above the maximum VFR authorized altitude.

The entire validation of the flight director and the study of the results for different pilots was divided into the following flight tests:

Flight test 1: Pilot: *Guillem Olivella*. Number of waypoints: 1. Goal of the flight test: First validation of the algorithm. Check that the algorithm is capable of showing commands to arrive at a waypoint at a concrete altitude. Check that the pilot is capable of compiling with the displayed commands. Check that the vertical and horizontal profiles are similar to those predicted.

Flight test 2: Pilot: *Guillem Olivella*. Number of waypoints: 3. Goal of the flight test: Second validation of the algorithm. Check that the algorithm is capable of showing commands to arrive at 3 consecutive waypoints at a concrete altitude. Check that the transition of the active "next waypoint" is done correctly between two consecutive waypoints.

Flight test 3: Pilot: *Marc Sabaté*. Number of waypoints: 3. Goal of the flight test: Third validation of the algorithm. Check if the experience of the pilot flying the airplane influence the results. The workload and the stress of the pilot will also be analysed.

Flight test 4: Pilot: *Jaume Olivella (under Guillem Olivella supervision)*. Number of waypoints: 3. Goal of the flight test: Forth validation of the algorithm. Check if a totally inexperienced pilot can follow a vertical and horizontal profile following the instrument. The possibility of using the instrument for flight instruction will be analysed.

5.1 Flight test 1

The objective of this first test was to check the program stability and the correct outputs to arrive to a single waypoint. The chosen point was Martorell and an altitude of 3000 ft. The GPS position and altitude to monitor the flight path and vertical profiles were taken each 10 seconds and have been displayed in Figure 5.2.

Figure 5.2: Horizontal and vertical paths of Flight Test 1.

Both profiles (vertical and horizontal) have been depicted with blue lines. The waypoint has been marked with a black cross and the desired altitude at each point is represented with a green line. The red semicircle (together with the red cross) represents the LELL ATZ limits, inside which the flight path corresponds to a standard departure. The test can be considered started once the line is crossed.

It can be noted that when the pilot follows the flight director, a correction is made for both the directional and the vertical deviations and Martorell is crossed at the correct altitude. The horizontal path is stable, correct and the response doesn't overshoot at any place. The vertical profile, however is not stable when following the flight director: the vertical position of the airplane oscillate around the objective altitude. Has to be said, however, that the oscillations have an amplitude of 30-50 ft which can be associated to small turbulences or GPS inaccuracies.

5.2 Flight test 2

The second flight test, executed by the same pilot as the first one, was designed to verify the algorithm responsible of changing the target point coordinates and altitude once the previous waypoint was crossed. It consisted of three waypoints at different altitudes: 1. Sant Sadurní d'Anoia at 3000 ft, 2. Vallbona d'Anoia at 3500 ft and 3. Sant Esteve sesrovires at 3000 ft. The resultant flight path and vertical profiles can be observed in Figure 5.3. Where the three waypoints are marked with black crosses and the objective altitude at each point is depicted in green.

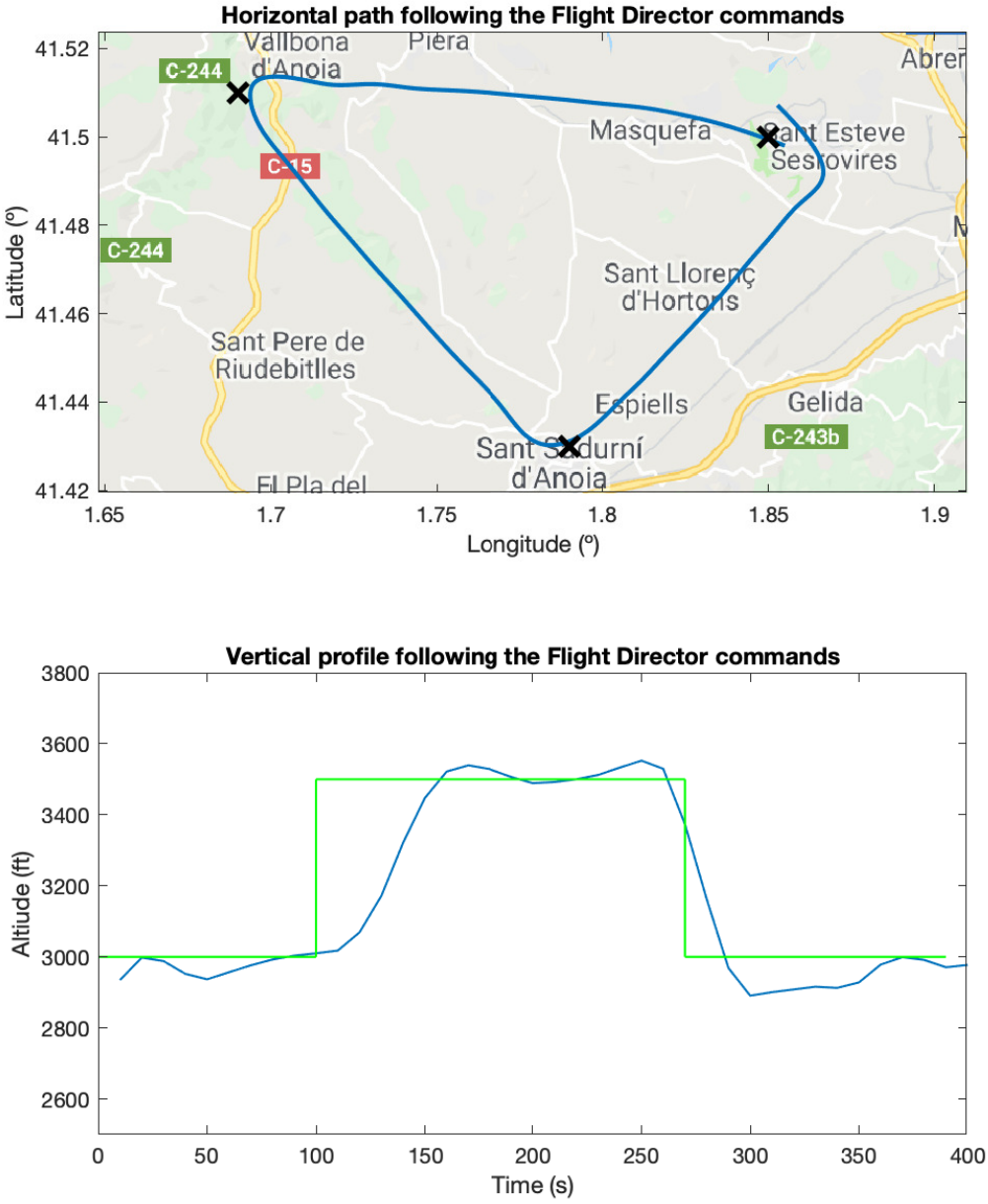


Figure 5.3: Horizontal and vertical paths of Flight Test 2.

Has to be remembered that to monitor the moment where the change of the target point has to be done, the flight director computer calculates the distance between the aircraft and the next waypoint; if this distance is smaller than a certain value, the objective waypoint (the next waypoint to be crossed) changes. The distance, below which the algorithm decides that the target point has to be changed, plays an important role in the flight director performance: if this value is very big, the airplane will not fly a strict fly-over but a fly-by route (this means that the waypoint will not be crossed). On the other hand, if the distance is very small, the airplane cannot detect either if the point is crossed or not; given that the position detected by the flight director can have inaccuracies. A first value of 1000 m was selected to do the flight test.

As it can be seen, the transitions between the waypoints are executed although Sant Sadurní d'Anoia and Vallbona d'Anoia are not overflown. This means that the value taken to change the target point is too big for a correct execution of the flight. Has to be said that the turn after crossing Sant Sadurní d'Anoia is bigger than the turn after crossing Vallbona d'Anoia. This is due to the saturation of the bank angle command when ascending (determined by the flight director parameter K). A limitation that does not occur when descending.

Regarding the vertical path, in general terms the pilot followed the predicted profile. However, oscillations around the desired altitude can be observed. As it has been explained previously, this may be caused by an instability of the control action, a bad reception of the GPS signal or perturbations caused by turbulences. During the flight, no oscillations of the pitch commands were perceived; it has been concluded, then, that the fluctuating altitude data is caused by either meteorological conditions, inaccuracies of the GNSS data or a combination of both.

The pilot tried to follow the RPM regime for which the flight director has been programmed: 2100 RPM in descent and 2400 RPM otherwise. This demand, together with the flight director commands (pitch and roll) constituted a stressful workload. When trying to maintain the attitude of the airplane, the engine RPM diverted from the correct value; and the same happened the other way around. A further analysis of this point will be done eventually.

Corrections to be done for the next Flight Test

1. The distance to the waypoint, below which the target point will be changed, will be reduced from 1000m to 500m.

2. A special attention will be put on the pilot workload when following the flight director.

5.3 Flight test 3

The third Flight test was executed by an experienced pilot ($>$ 600 h flight time) to evaluate the flight path and vertical profile when the flight director was followed with high precision. And to analyse to workload experienced by the pilot. As in the previous cases, together with the attitude commands displayed by the application, the engine regime was maintained at the predefined values when performing an ascend, levelled flight or descent. The result is displayed in Figure 5.4.

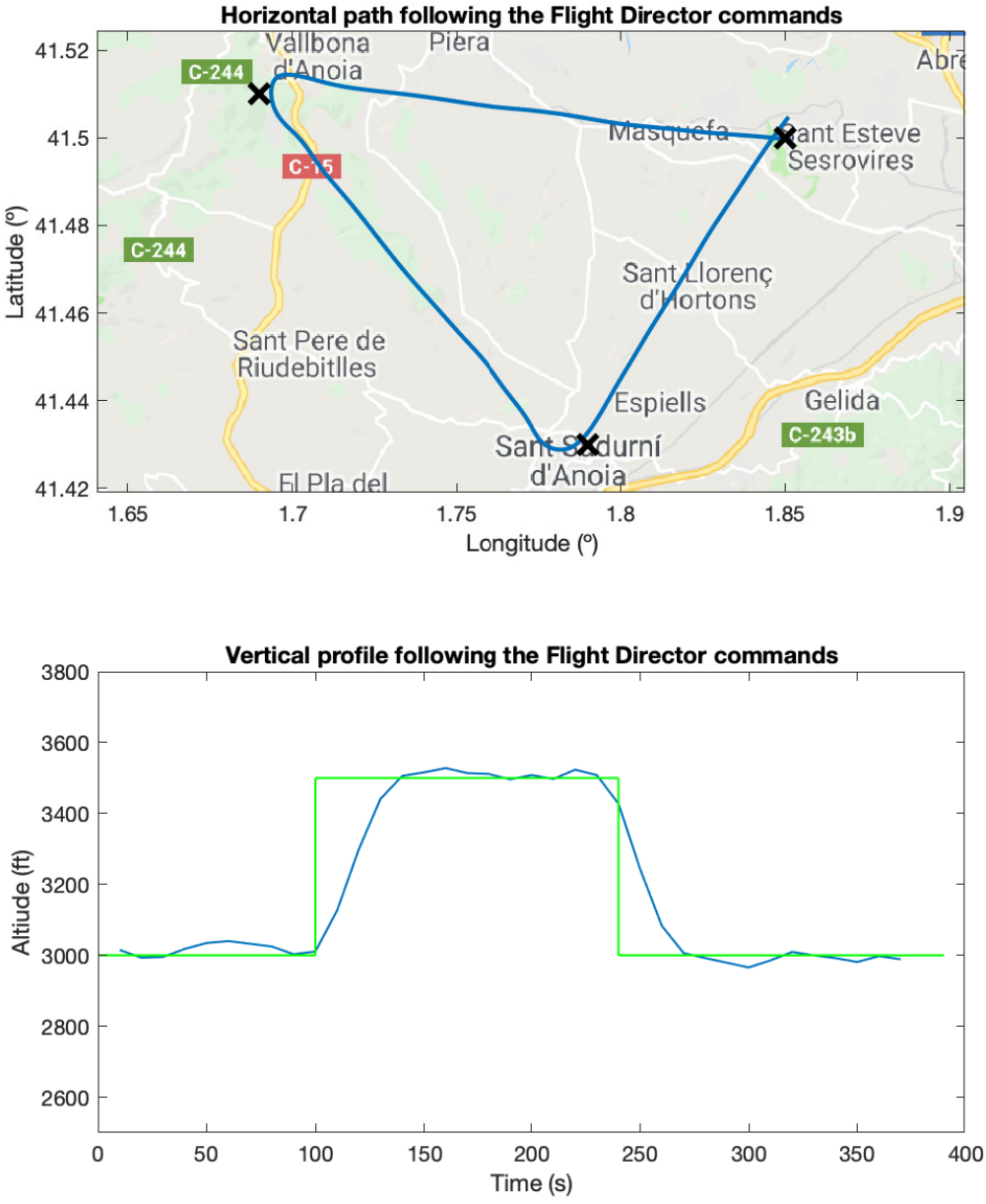


Figure 5.4: Horizontal and vertical paths of Flight Test 3.

Both the horizontal path and the vertical profile were efficient, exact and the pilot managed to maintain the airplane in the desired state at each time. The legs joining the different waypoints are straight and no big corrections where needed. The difference in the turn radius when rolling above the first and second waypoint cannot be appreciated. However, the bank angle was smaller when performing the ascent manoeuvre: this can be associated to the fact that when ascending, the velocity was smaller and, for a different bank angle, the turn radius was the same. For the vertical profile, the transitions between the ascent or descent states and the levelled flight was done in a smooth way, without big overshooted results.

As in the previous flight tests, the transition between the waypoints was successfully done. It can also be seen that, by reducing the distance from the aircraft position and the next waypoint, below which the target point changes, the route resembles more a fly-over than a fly-by route.

The pilot revealed that following the flight director was not a complicated task but having to maintain the engine regime at a predefined level constituted to many requirements to be followed simultaneously.

5.4 Flight test 4

The aim of this last flight test was to evaluate the flight director with a totally inexperienced pilot. The subject had never piloted an airplane before and he was instructed to follow the flight director commands together with the throttle to maintain the required RPM. During all the flight, an experienced pilot was controlling the flight status for security reasons. The commanded flight path was composed by the same waypoints at the same altitude and the result can be appreciated in Figure 5.5.

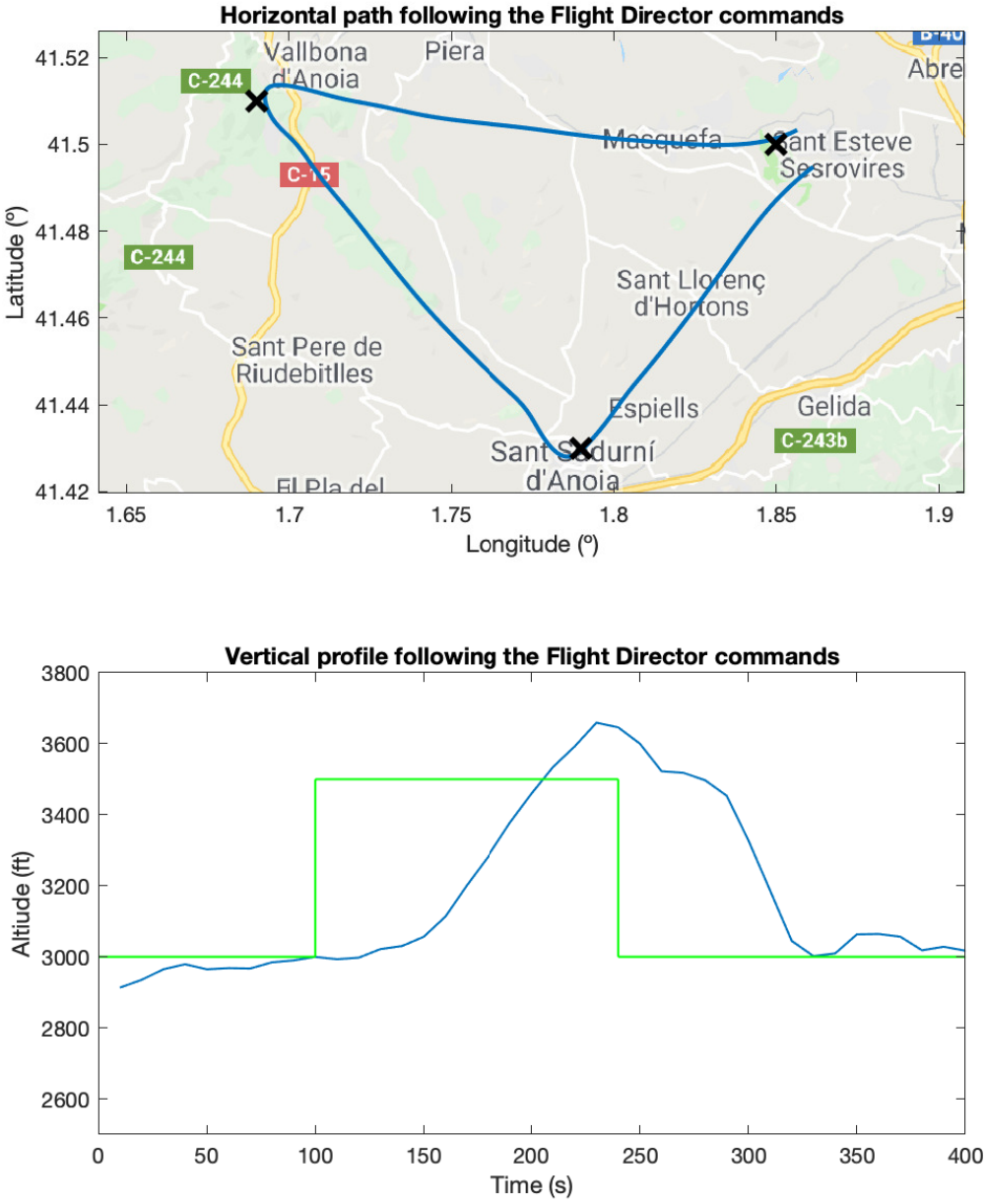


Figure 5.5: Horizontal and vertical paths of Flight Test 4.

The flight executed by the inexperienced pilot revealed that he couldn't follow the flight director commands simultaneously. When the pilot's attention was focused on the roll correction, he couldn't manage to deal with the pitch and engine RPM instructions. In general terms, the result was a correct horizontal flight that passed through the selected waypoints in an efficient way. However, given by deviations in the engine regime and pitch requirements, the vertical profile was very different from the predicted one. When doing the transition from 3000 ft to 3500 ft, a delay in the altitude response can be observed. This is due to the fact that the pilot was only concerned about the lateral correction to be done to the aircraft. Another fact evidencing the incapacity of the pilot to follow the pitch command is that when trying to reach the desired altitude, the vertical profile shows an overshooted response. Has to be said that during the flight, important variations on the required engine regime were observed, too. Moreover, when the pilot followed the instrument commands, he only obeyed partially: the pitch and roll transmitted to the aircraft were smaller than the value displayed by the flight director.

The pilot also showed a stressful attitude because of the workload of the task. Has to be said, then, that using this instrument for initial phases of flight training may not be an optimum method to learn how the airplane reacts when applying certain inputs.

As happened during Flight test 3, the transitions between the waypoints have been done correctly and the target points changed relatively near the waypoint vertical. Although the fly path is not completely a fly-over route, it is very similar to it. In Chapter 7, other techniques and methodologies to improve this point will be discussed.

Regarding the horizontal path, the inexperienced pilot didn't follow the exact bank angle commanded by the instrument and, consequently, the differences in the turn radius cannot be appreciated. In fact, a steeper turn was done in the first waypoint compared to the turn in the second one.

6 | Discussion and Conclusions

Having obtained the flight tests results and the pilots feedback, a final analysis of the instrument developed can be done. On the other hand, the environmental impact and the security study of the design will be discussed.

6.1 Design evaluation and results analysis

Evaluating the two first flight tests, it can be said that the flight director managed to guide the pilot through the desired waypoints at the correct altitude. It can also be said that during all the stages of the flight the engine had enough power to supply the commands of pitch and roll given by the flight director. The hypothesis applied to the physical model have not implied a reduction in the accuracy of the system nor the Simulink flight simulator. On the other hand, some pilotage conditions (of velocity and engine RPM) have been necessary to control the aircraft state through the flight. These demands, generally of maintaining a predefined engine regime, apart from the flight director commands of pitch and roll, showed to increase the pilot workload: when the flight director designed wanted to be a help to the pilot, the method developed can suppose an increase of work when piloting.

The most important difference between the commercialized flight directors and the one designed in this project is the capacity of interacting with the airplane systems. When the other flight directors can read and interpret the instrumentation on board of the airplane, the application developed is totally independent. Moreover, systems as GARMIN G1000 or Boeing or Airbus families have an autothrottle associated with the flight director which reduces, significantly, the pilot demands. Regarding the design, although the flight director has achieved the objectives proposed (crossing certain points at certain altitudes), the flight path (horizontal and vertical profiles) was not totally straight because of velocity and engine RPM unwanted variations.

For the application development, different alternatives have been evaluated in multiple stages of the design and the most appropriate option has been selected in each case. The justifications for the elections have been based on parameters involving: cost, time and complexity of implementation. Regarding the use of the flight director for navigation, the application has some limitations that need to be marked: In the first place, when executing a flight in VFR and single pilot, the utilization of the instrument may not be safe because of a possible lost of the sight of view of nearby traffics; or a possible insecure proximity to the clouds. The algorithm itself also has some limitations: Firstly, if two waypoints are near in space and their altitude is also very different, the flight director would not be able to display a pair of pitch and roll commands to arrive to the target waypoint. Although the limitation is given by the aircraft dynamics, the instrument does not warn the pilot about the problem. Moreover, and since the hardware is independent of any other system, no navigation back-up exists if the system may fail: given, for example, by a loss of the GPS signal. In fact, the

accurate operation of the instrument is conditioned by a good GNSS reception.

The differences in the horizontal path and vertical profile followed by the different pilots with different flight hours revealed that the most experienced pilot managed to reach the waypoints in a more stable and efficient way. The horizontal path, however, was an easier task to follow for the inexperienced pilot. The vertical profile, which was the result of the combination of pitch and engine regime commands, was only successfully followed by the most experienced pilot. The knowledge of the aircraft dynamics and its reaction to a given input, resulted in a quicker response to the flight director demands. The consequence of these facts were bigger delays and overshooted pitch and roll attitudes when trying to follow the instrument. However, both pilots coincided on the fact that following both: the flight director commands and the RPM needed in each stage of the flight was a stressful task.

To sum up, it can be concluded that an optimum way to resolve the problem, defined in the aim of the project, has been found. However, the accuracy of some design parameters and methods used in the main algorithm are to be improved. The limitations of the hardware and the software have been identified and ways to improve them explained. Has to be said that, although many authors identify improvements in flight directors as a potential avenue to increase pilot performance [28], it has been concluded that a flight director independent of the aircraft system is not a viable solution to reduce the pilot workload when flying. The interactions between the airplane and the commercial flight directors are two-way connections: in the first place, the flight director reads the aircraft instruments (in order to increase the accuracy in its calculations); and, in the second place, the flight director runs the autothrottle to reduce the pilot workload. Without any of this two interactions, the navigational instrument loses utility, efficiency and precision.

6.2 Environmental impact

The environmental implications of this project are not obvious since the study has been done with the aim of developing the theory behind an hypothetical flight director in an academical framework. Although the impact on the environment cannot be estimated, it is true that the implementation of more advanced navigation systems (such as flight directors) on modern airplanes would increase the efficiency when travelling from one place to another. And, consequently, fuel would be reduced as well as a reduction of pollutants in the atmosphere.

6.3 Security evaluation

Regarding the security, when executing the flight test, precautionary measures have been taken to assure a safe flight: One example of this is the fact of having two pilots on board and checking periodically certain parameters using checklists. On the other hand, the developed application has a warning message to prevent the user from using it as the main navigation method.

7 | Future improvements

In this section, the possible improvements to be done on this project to upgrade the prototype will be analysed and the best way to achieve them will be discussed. On the first hand, from a commercial point of view, the iOS application could be extended to make it more user-friendly and the program could be exported to other operative systems. One possible improvement would be to use a Google Maps API, to select the waypoints on the route and their respective altitudes. By doing so, the application could be used with different routes at any time and reroutings could be done on board.

Another improvement that could be done to the prototype to improve it, would be to adjust the control algorithm to the weight of the aircraft. In this project, the aircraft mass has been considered constant and known (obtained by adding the operational empty weight of the aircraft and the weight of the fuel). The way to do this would be to identify the dependence of the parameters to the aircraft weight and adjusting them prior the beginning of the flight by asking the pilot to introduce the aircraft mass in the application.

Although it has not been considered in the scope of this project, the flight director could be extended to other applications apart from en-route operation. An example, would be to reprogram the application to follow a VOR radial. To do so, a different path-following method will be needed: any algorithm considering initial and final waypoints in a leg would be valid (two examples would be the carrot-chasing algorithm or the non-linear law).

Another improvement path would consists on increasing the precision of the parameters modelling the airplane and retuning the variables of the control action. To develop the controller, a set of data obtained from the bibliography, by heuristic methods or by a combination of of both has been used to adjust the values defining the flight director algorithm. In table 7.1, this parameters are identified and possible ways to improve its precision explained. A way to continuously adjust the algorithm data, would be to design an open-source application where different pilots or engineers could upload their design parameters in a predefined form. By doing so, the application would then be extended to other aircrafts of the same type (single engine fixed pitch propeller aircraft). However, in order to incorporate other types of airplanes (such as jets or variable pitch propellers) the model defining the dynamics of the aircraft should be changed to readjust the curve of maximum pitch and roll as a function of K.

Data	**Possible ways of improving precision**
Aircraft parameters: Wing Area (S), Aircraft TOM (m), $C_l(\alpha)$, $C_d(\alpha)$: ($C_{l\alpha}$, C_{l0}, C_{d0}, k)	Some aircraft parameters such as the weight or the wing surface have already been determined with a high precision. However, other parameters - such as the aerodynamic coefficients - could be corrected by doing appropriate experiments or flight tests.
Pilot parameters: reaction time (σ and μ)	The reaction time has been used to check the stability of the control algorithm when applying a delay caused by the pilot action. A better determination of the distribution of reaction times of general aviation pilots, could improve the study of the limitations of the control action.
Control parameters (1): Maximum pitch and roll as a function of k (parameters A and B)	The curve representing the maximum pitch and roll attitudes of the aircraft as a function of the relative position to the following point has been determined using experimental methods. And A and B have been determined by approximation. More points and repetitions could have improved the results and the performance of the control action.
Control parameters (2): K_p, K_{pA}, K_{dA}, N, K_{iA}	The control parameters of the two PIDs, have been calculated by heuristic methods. By correcting all the data mentioned above, new values could be obtained and the performance of the flight director could be improved.
Control parameters (3): Maximum and minimum absolute pitch and roll	The 4 values have been obtained from the aircraft dynamic model analysis. By correcting the values defining it, new results could have been obtained and the saturation values of the flight director redefined.

Table 7.1: Data to be improved and possible ways to do so

A conflictive point of the algorithm has been the condition imposed to know when to change the target point in a sequence of waypoints defining a path. To resolve the problem, it has been supposed that a fly-over route can be considered a fly-by horizontal path where the distance to the following waypoint, below which the target point could be changed to the next waypoint in the route was very small. This method produced some errors and the route flown was not exactly a fly-over route. Two alternatives to resolve this point of the algorithm will be given:

The first option will be to wait a certain time after the algorithm detects that the distance from the airplane to the next waypoint is below a predefined value. By doing this, if the calculation of this time is done correctly and a security margin is included, the airplane will overfly the desired waypoint and after doing so, it will change the target point to the following location in the route.

The second option would consists on detecting when the airplane crosses an imaginary segment perpendicular to the leg joining the previous and the next waypoint. By doing so, the algorithm will only change the target point if the airplane crosses, without any doubt given by estimations that can lead to erroneous results, the waypoint selected. This solution is more difficult to implement because of the spherical coordinates used in the navigation system. A scheme of this solution is presented on Figure 7.1.

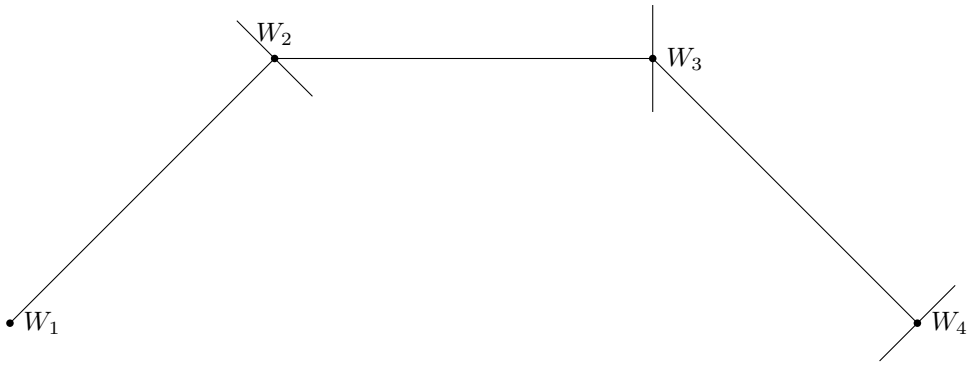


Figure 7.1: Second method to detect when to change the target point.

One of the problems of this method is that the airplane will follow the waypoint until it will be very close to it. This can induce ζ to become very large, which consequently can make the algorithm display undesired big bank angle commands.

7.1 Future work programming

In table 7.2, the amount of hours that may be needed to improve the project, by doing certain tasks, is predicted. Has to be said, however, that many other paths can be followed to improve the study and design of flight directors in VFR, such as a complete different approach to the problem.

Future improvement	**Task**	**Time prediction**
Export the program to other operative systems: Native application (Android), hybrid program	Design	100 h
	Implementation	250 h
Capability of selecting waypoints inside the application by using a Google Maps API	for Android	150 h
	for iOS	150 h
Improve the performance of the algorithm by increasing the accuracy of its parameters	Parameters dependent on the aircraft mass	250 h
	Increase the precision of other parameters	50 h
Extend application to the path following techniques (Intercepting VOR radials, ...)	Study viability	50 h
	Implementation	150 h
Extrapolating application to other airplanes	Fix-prop single engine	100 h
	Other types	200 h
Implementing other methods to change the target point in a sequence of waypoints.		100 h

Table 7.2: Prediction of hours needed to improve the project by doing certain tasks

Bibliography

[1] Khoury, A.; Gillett, J. *Airship Technology*. Cambridge, United Kingdom: Cambridge University Press, 1999.

[2] G. Iddings, E. Martino, "Flight Director Design Trends", *IRE Transactions on Aeronautical and Navigational Electronics*, vol: ANE-2, no. 1, March, 1955.

[3] G1000, *Integrated Flight Deck Pilot's Guide for Cessna Nav III*, GARMIN, 2011.

[4] Crane, Carl J., "Aircraft flight control optical indicating device", U.S. Patent 2685226A, 1950.

[5] K. P. Katz and J. L Ozimekt, "Integrated Digital Avionics to Improve Aircraft Environmental Control Systems" *J. AIRCRAFT*, vol. 33, no. 1, pp. 238-240.

[6] U.S. Department of Transport (2009). *Advanced Avionics Handbook*. Washington D.C., EEUU: Federal Aviation Administration. p 4-4.

[7] Flight Crew Operations Manual, 737-6Q8/-7Q8/-8Q8. *MALEV Hungarian Airlines*, 2005.

[8] Flight Crew training Manual, A318, A319, A320, A321. *Airbus*, 2008.

[9] Cessna 152 Pilot's Operating Handbook And FAA Approved Airplane Flight Manual. *abridged for KCN Aero Club*, 2003.

[10] Easy acces rules for Standarised European Rules of the Air (SERA), *European Aviation Safety Agency*, 2018.

[11] U.S. Department of transportation, "Civil utilization of area navigation (RNAV) departure procedures", Federal aviation administration, 2001. [Online]. https://www.faa.gov/documentLibrary/media/Order/ND/8260.44A%20Chg1.pdf.

[12] G. Conte, S. Duranti, and T. Merz, "Dynamic 3D path following for an autonomous helicopter," *IFAC Proceding Volumes*, vol. 37, no. 8, 2004, pp. 472–477.

[13] P.B. Sujit, S. Saripalli, J. B. Sousa, "Unmanned aerial vehicle path following", *IEEE Control systems magazine*, pp. 42 - 59, february 2014.

[14] S. Park, J. Deystt, and J. P. How, "Performance and Lyapunov stability of a nonlinear path-following guidance method," *J. Guidance, Control, Dyn.*, vol. 30, no. 6, pp. 1718–1728, 2007.

[15] G. Miquel, P. Manuel, P. César, *Mecánica del vuelo*. Garceta, Madrid, 2012.

[16] J. Scott, Drag Coefficient & Lifting Line Theory, 11 July, 2014, [Online]. http://www.aerospaceweb.org/question/aerodynamics/q0184.shtml.

[17] I. Martín, M. Asunción. *Trigonometría esférica : teoría y problemas resueltos*. Universidad del País Basco, 2004.

[18] Klein, R. H., McRuer, D. T., and Weir, D. H., "A Pilot-Vehicle Approach to Longitudinal Flight Director Design", *Journal of Aircraft*, vol. 8, no. 11, 1971, pp. 890–897.

[19] A. Zulu and Samuel John, "A Review of Control Algorithms for Autonomous Quadrotors", *Open Journal of Applied Sciences*, no. 4, 2014, pp. 547-556.

[20] M. Vladimir, E. Vivas and C. Rodríguez, "Simulation of the quadrotor controlled with LQR with integral effect", *ABCM Symposium Series in Mechatronics*, vol. 5, pp. 390 - 399.

[21] B. Masselink, M. Mulder, A. in 't Veld and M. M. Van Paassen, "Design and Evaluation of a Flight Director for Zero and Partial Gravity Flight", Conference: AIAA Guidance, Navigation, and Control Conference, August, 2009.

[22] J. Boril and R. Jalovecky, "Parameter Estimation of Transfer Function of Pilot Behaviour Model from Measured Data" Recent Researches in Automatic Control, Systems Science and Communications, WSEAS Press, Portugal, 2012, pp. 144-148.

[23] G. Welch and G. Bishop, "An Introduction to the Kalman Filter", *University of North Carolina*, Chapel Hill, 2014.

[24] Apple Developer, "Framework Core Location", 2019. [Online] https://developer.apple.com/documentation/corelocation.

[25] L. Rundqwist, "Anti-reset Windup for PID Controllers", *IFAC Proceedings Volumes*, vol 23, no 8, pp 453-458, August 1990.

[26] M. Kells, F. Willis, J. Bland, *Plane And Spherical Trigonometry*, McGraw Hill Book Company, Inc. pp. 323–326, 2004 (Retrieved July 13, 2018).

[27] R. Sinnott, "Virtues of the Haversine", *Sky and Telescope* vol. 68, no. 2, pp. 159, 1984

[28] S. De Stiger, M. Mulder, and M. Van Paassen, "Design and Evaluation of a Haptic Flight Director," *Journal of Guidance, Control and Dynamics*, vol. 30, no. 1, 2007, pp. 35–46.

A | Flight path Simulink simulation

This appendix contains the Simulink model that has been used to simulate the pilot - airplane - flight director interaction. Apart from the block diagrams and programs constituting the simulation, a scheme of the model hierarchy is shown as well as the different numerical values used.

A.1 Full Model Hierarchy

1. Pilot
2. Airplane Model
3. Kinematic Model
4. Waypoint Selector
   4.1. CalculateNextWaypoint
5. Flight Director
   5.1. Subsystem 1
   5.2. Subsystem 2
   5.3. Angle converter
   5.4. Roll Limiter
   5.5. Theta Limiter

A.2 Model variables

Two types of model variables have been distinguished: the ones related to the navigational parameters (such as initial locations of the aircraft), control actions (PID gains) and aircraft dynamic and geometric data. And a second type related to the curves representing the maximum and minimum pitch and roll commands that can be displayed on the flight director. This second type of data consists of three vectors representing: (1) a sorted list of values that could be adopted by parameter K, (2) a list of the pitch angles corresponding to the mentioned values of K and (3) a list with the roll angles also corresponding to the different values of vector (1). Because of the magnitude of the three vectors, they are not shown in this appendix.

A.2.1 Navigational, control and aircraft parameters

Variable Name	**Definition**	**Value (if fixed)**
A	$\frac{1}{2} \cdot \rho \cdot S$ (kg/m)	9.126
Anglew	Wind direction in rad (with respect to True North)	-
B	$m \cdot g$ (N)	7426
Cd0	Minimum drag coefficient (Aerodynamic parameter)	0.027
Chi_0	Initial heading in rad (with respect to True North)	-
Cl0	Lift coefficient at 0 angle of attack	0.350
Cldmin	Lift coefficient at minimum drag coefficient	0.000
Clslope	Lift coefficient slope in rad^{-1}	4.77
KdA	Altitude control PID: derivative control parameter	-0.008
KiA	Altitude control PID: integrator control parameter	0.0003
Kp	Direction control PID: proportional control parameter	0.9
KpA	Altitude control PID: proportional control parameter	0.004
N	Altitude control PID: discrete-time derivative parameter	0.1
Re	Earth radius in metres	6371000
V	Airspeed velocity in metres per second	-
Vaw	Upward current of air velocity in metres per second	-
Vw	Wind velocity in metres per second	-
g	Gravity in metres per second squared	9.81
h_0	Initial altitude in metres	762
k	Second order drag coefficient parameter	0.07
lat_0	Initial latitude in rad	0.724
long_0	Initial longitude in rad	0.0336

A.3 Generic Model Block Diagram

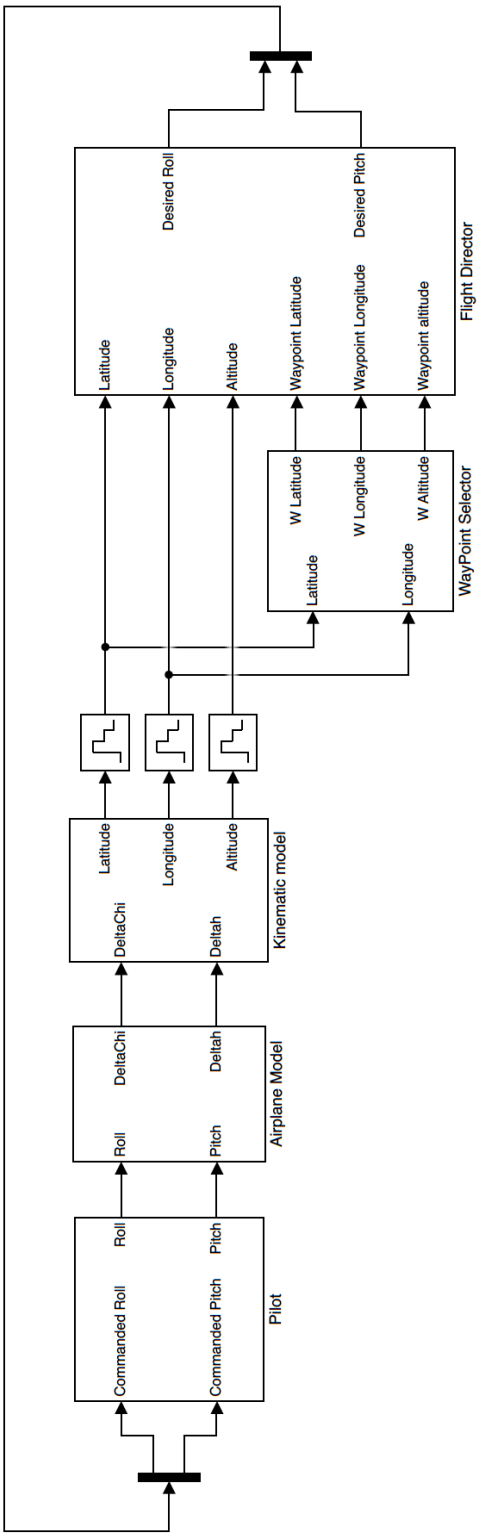


Figure A.1: Generic Simulink Block Diagram

A.4 Model/Pilot

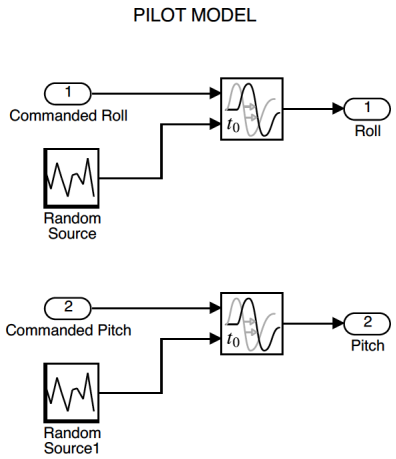


Figure A.2: Pilot Simulink Block Diagram

Mean of the roll command reaction time	0.6 s
Variance of the roll command reaction time	0.1 s^2
Mean of the pitch command reaction time	0.6 s
Variance of the pitch command reaction time	$0.1s^2$

Table A.1: Pilot reaction time numerical data

A.5 Model/Airplane Model

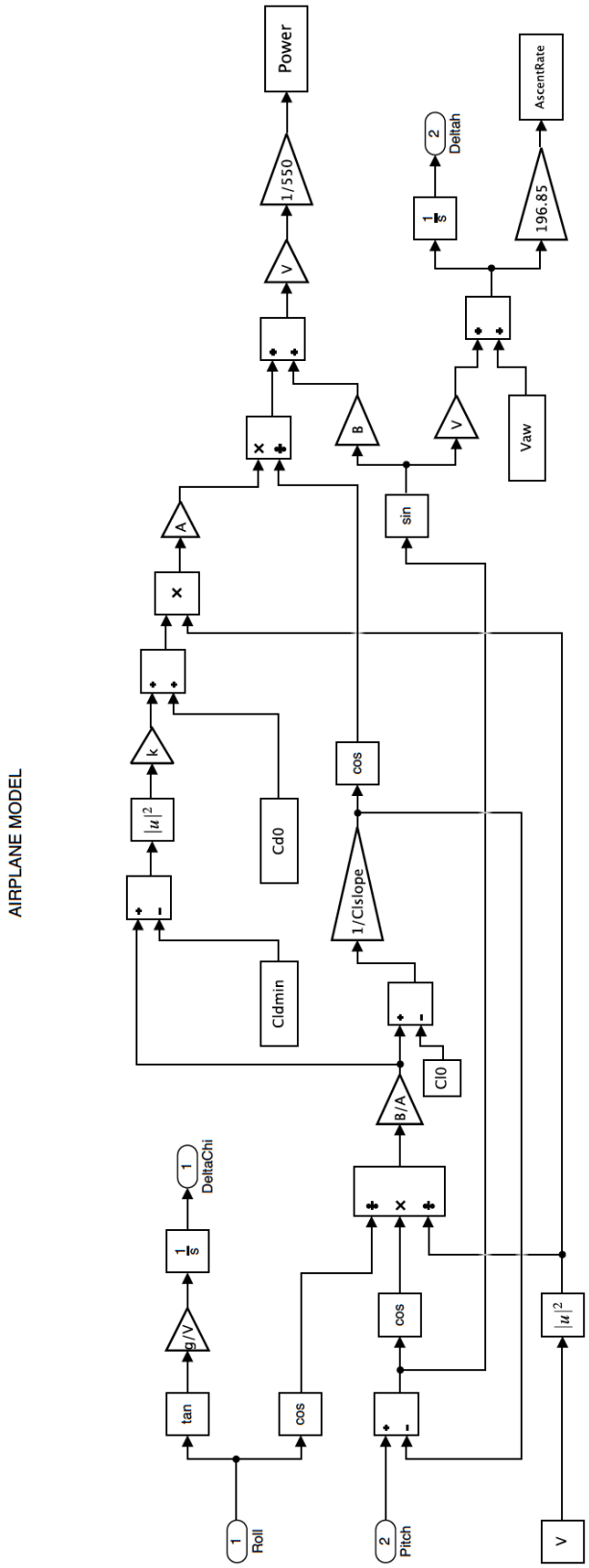


Figure A.3: Airplane Model Simulink Block Diagram

A.6 Model/Kinematic Model

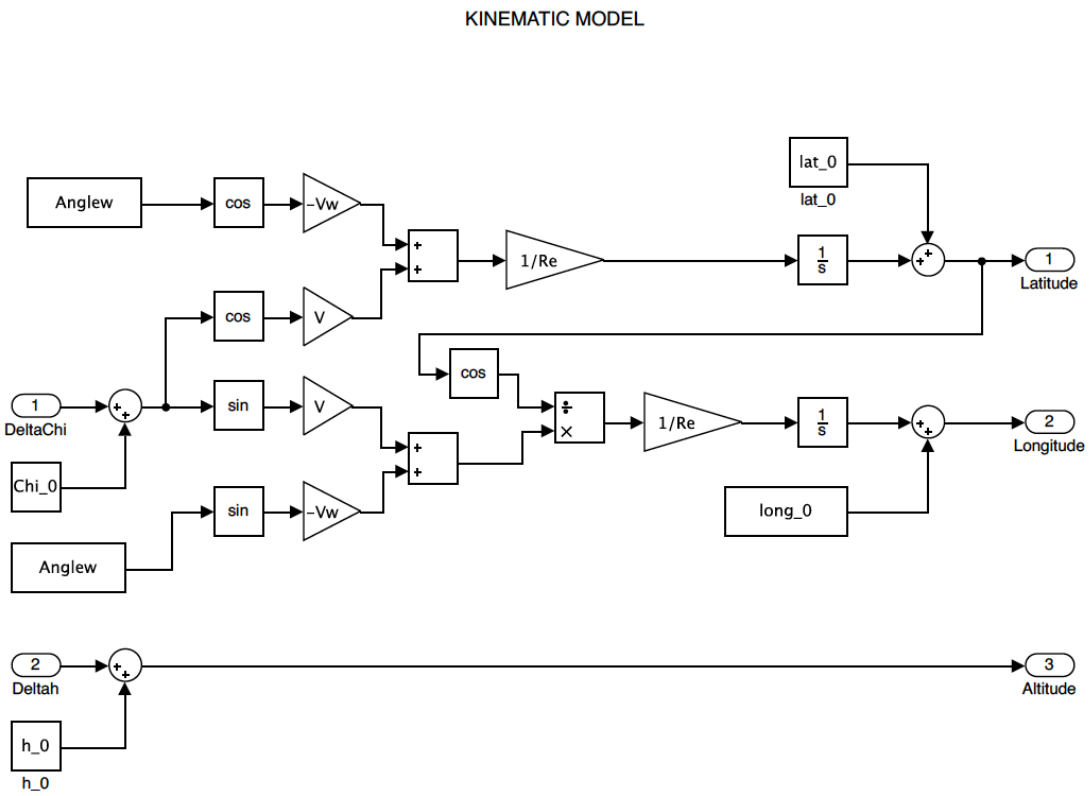


Figure A.4: Kinematic Model Simulink Block Diagram

A.7 Model/Waypoint Selector

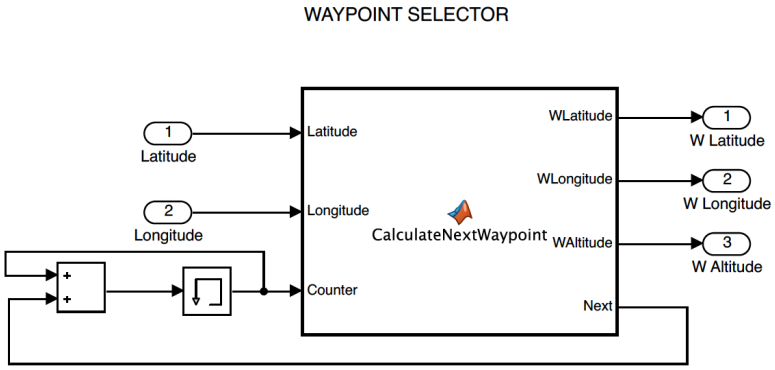


Figure A.5: Waypoint selector Simulink Block Diagram

A.8 Model/Waypoint Selector/ CalculateNextWaypoint

```
function [WLatitude, WLongitude, WAltitude, Next] = ...
    CalculateNextWaypoint(Latitude, Longitude, Counter)
    % DATA
    WaypointLatitudes = [41.42/180*pi, 41.51/180*pi, 41.50/180*pi];
    WaypointLongitudes = [1.79/180*pi, 1.69/180*pi, 1.85/180*pi];
    WaypointAltitudes = [3000*0.3048, 4500*0.3048, 3500*0.3048];
    % CONTROL PARAMETERS
    Next = 0;
    R = 6371000;
    dChange = 100;
    if Counter < 1
        Counter = 1
    end
    WLongitude = WaypointLongitudes(Counter)
    WLatitude = WaypointLatitudes(Counter)
    WAltitude = WaypointAltitudes(Counter)
    LatitudeLocal = Latitude (length(Latitude))
    LongitudeLocal = Longitude (length(Longitude))
    % 1. Distance computation between waypoint and current position
    dlon = WLongitude - LongitudeLocal;
    dlat = WLatitude - LatitudeLocal;
    a = sin(dlat/2).*sin(dlat/2) + cos(WLatitude).*cos(LatitudeLocal) .* ...
    sin(dlon/2).*sin(dlon/2);
    c = 2 * asin(min(1,sqrt(a)));
    d = R * c
    % 2. Check next waypoint
    if (d < dChange && Counter<length(WaypointLongitudes))
        Next = 1;
        WLongitude = WaypointLongitudes(Next);
        WLatitude = WaypointLatitudes(Next);
        WAltitude = WaypointAltitudes(Next);
    end
end
```

A.9 Flight Director

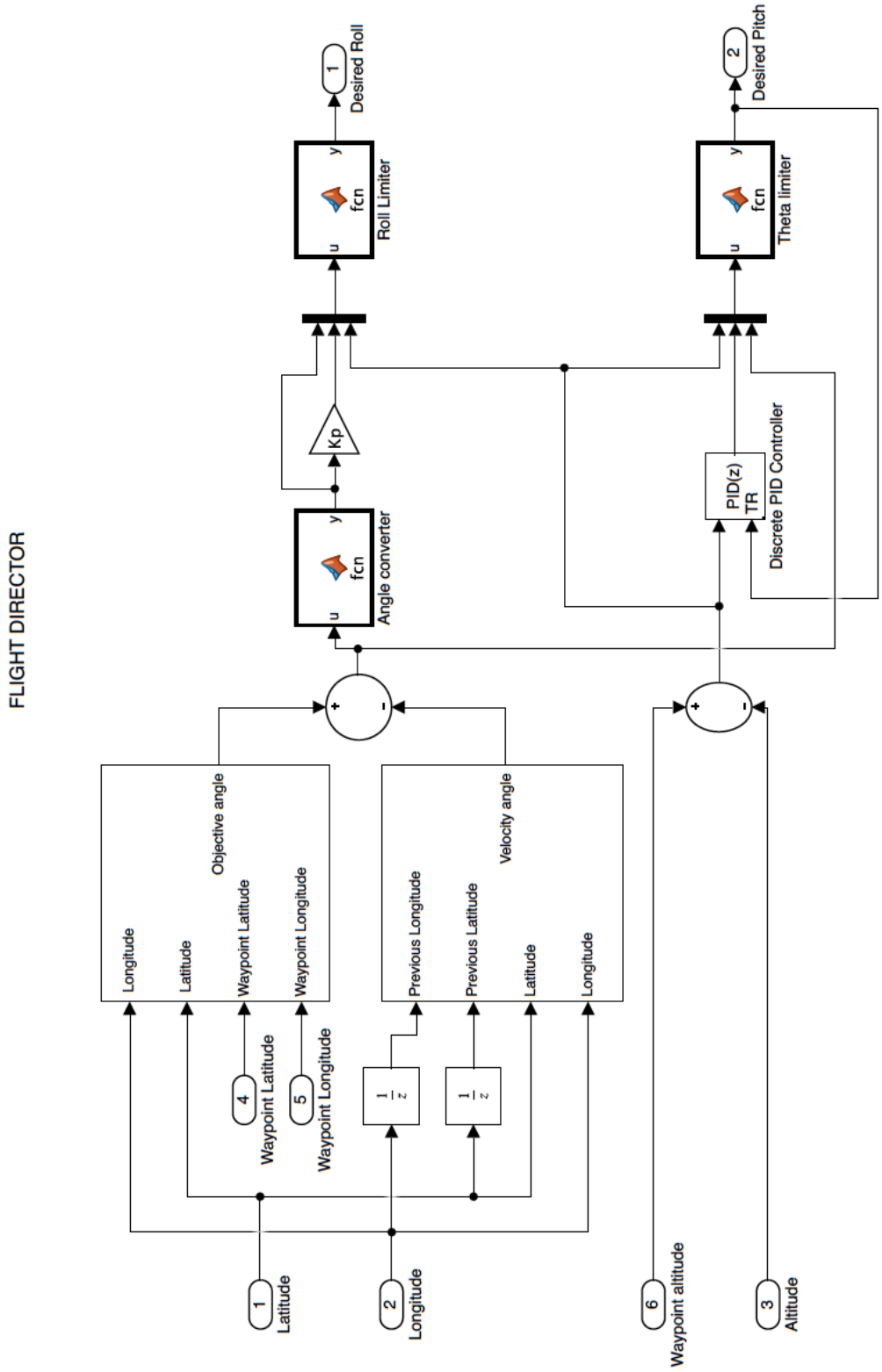


Figure A.6: Flight Director Simulink Block Diagram

A.10 Model/Flight Director/Angle Converter

```
function y = fcn(u)
    y = u
        if y > pi
            y = y - 2*pi
        elseif u < -pi
            y = 2*pi + y
        end
end
```

A.11 Model/Flight Director/Roll Limiter

```
function y = fcn(u)
    polimu = spline(invfrac,invmu)
    u1 = u(1);
    u2 = u(2)/pi*180;
    u3 = u(3);
    if u1 == 0
        y = u2;
    elseif u3<0
        if u2<-17.7
            y = -17.7;
        elseif u2>17.7
            y = 17.7;
        else
            y = u2;
        end
    else
        prop = u3/u1;
        if prop>6647
            prop = 6647
        elseif prop<0.73 && prop>0
            prop = 0.73
        elseif prop<-6647
            prop = -6647
        elseif prop>-0.73 && prop<0
            prop = -0.73
        end
        ValorMu = ppval(polimu,abs(prop));
        if abs(u2)>ValorMu
            if u2>0
                y = ValorMu;
            else
                y = -ValorMu;
            end
        else
            y = u2;
        end
    end
    y = y/180*pi;
end
```

A.12 Model/Flight Director/Theta Limiter

```
function y = fcn(u)
    politheta = spline(invfrac,invtheta);
    u1 = u(1);
    u2 = u(2)/pi*180;
    u3 = u(3);
    if u3 == 0
        y = u2;
    else
        prop = u1/u3;
        if prop>3000
            prop = 3000
        elseif prop<0.73 && prop>0
            prop = 0.73
        elseif prop<-3000
            prop = -3000
        elseif prop>-0.73 && prop<0
            prop = -0.73
        end
        ValorTheta = ppval(politheta,abs(prop))
        if u2>ValorTheta && u2>0 && abs(u1)>30
            y = ValorTheta;
        elseif u2<-2.5
            y = -2.5
        else
            y = u2
        end
    end
    y = y/180*pi
end
```

A.13 Model/Flight Director/Subsystem 1

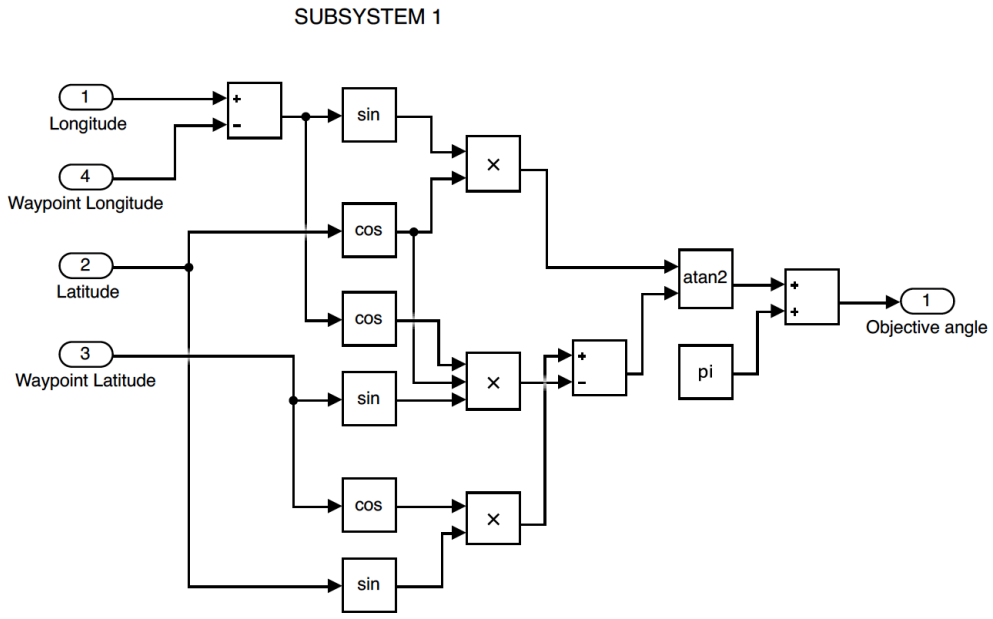


Figure A.7: Subsystem 1 Simulink Block Diagram

A.14 Model/Flight Director/Subsystem 2

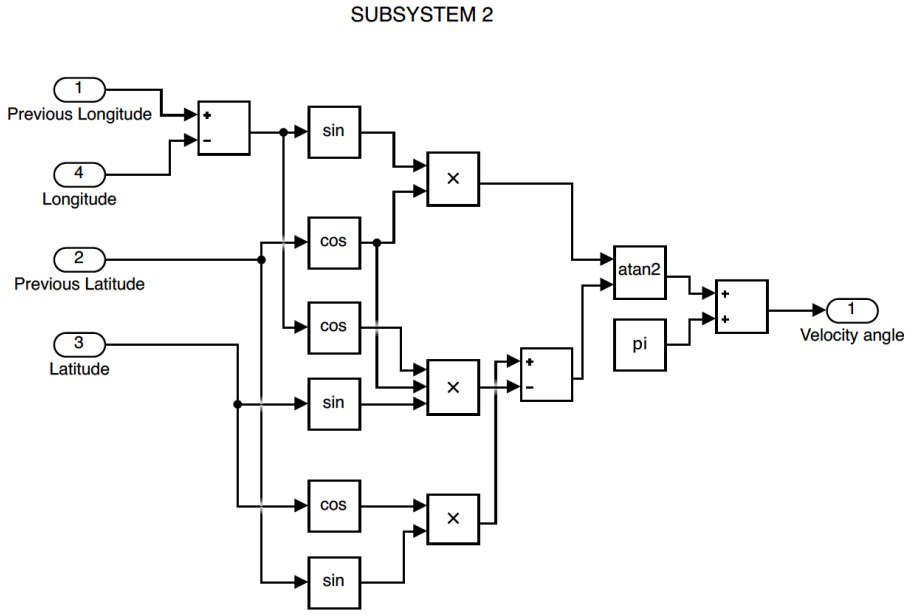


Figure A.8: Subsystem 2 Simulink Block Diagram

B | iOS application files

This appendix contains the Swift program that runs the iPhone application. The App works for any iOS device having GNSS capabilities. Two main files are shown: the first one containing the graphical user interface and the second one containing the main program running the control action involved with the flight director.

B.1 Graphical User Interface

```
1
2  //
3  //  ArtificialHorizon.swift
4  //  ArtificialHorizon
5  //
6  //  Created by Guillem Olivella i Marti on 18/11/2018.
7  //  Copyright 2018 Guillem Olivella. All rights reserved.
8  //
9
10 import UIKit
11 let WingSize = 100
12 let WingStart = 60
13 let WingWidth = 10
14 let PointSize = 45
15 let TriangleSize = 30
16 let PitchWidth = 200.00
17 let ArcFactor = 0.4
18 class ArtificialHorizon: UIView {
19     //Artifitial Horizon Bezier Path. Trigonometric relations have been taken ...
       into account.
20     override func draw(_ rect: CGRect) {
21         //Terrain
22         let Terrain = UIBezierPath()
23         pitch = -pitch-pi/2
24         let x = CGFloat(0)
25         let y = CGFloat(tan(-pitch)) * bounds.height / 2
26         let xcenter = bounds.width / 2 + (x * ...
       CGFloat(cos(roll))-y*CGFloat(sin(roll)))
27         let ycenter = bounds.height/2 - ( y*CGFloat(cos(roll)) + ...
       x*CGFloat(sin(roll)))
28         let centerPitch = CGPoint(x: xcenter, y: ycenter)
29         Terrain.move(to: centerPitch)
30         Terrain.addLine(to: CGPoint(x: bounds.width, y: ycenter + (bounds.width - ...
       xcenter) * CGFloat(tan (-roll))))
31         Terrain.addLine(to: CGPoint(x: bounds.width , y: bounds.height ))
```

```
32          Terrain.addLine(to: CGPoint(x: 0.0, y: bounds.height))
33          Terrain.addLine(to: CGPoint(x: 0.0, y: ycenter - xcenter * CGFloat(tan ...
        (-roll))))
34          Terrain.close()
35          UIColor.brown.setFill()
36          Terrain.fill()
37          //Pitch lines
38          let PitchSmallLine = UIBezierPath()
39          var Dist = 0.0
40          for i in stride(from: -60, through: 60, by: 5) {
41              if (Double(i)>pitch/pi*180-35 && Double(i)<pitch/pi*180+20){
42                  if (i % 10 == 5 || i % 10 == -5){
43                      Dist=PitchWidth/2
44                  }
45                  else {
46                      Dist=PitchWidth
47                  }
48                  let j = Double(i)*pi/180
49                  let x = CGFloat(Dist)
50                  let x2 = -CGFloat(Dist)
51                  let y = CGFloat(tan(-pitch+j)) * bounds.height / 2
52                  let PitchSmallCenter =  CGPoint(x: bounds.width / 2 + (x * ...
        CGFloat(cos(roll))-y*CGFloat(sin(roll))), y: bounds.height/2 - ...
        (y*CGFloat(cos(roll)) + x*CGFloat(sin(roll))))
53                  PitchSmallLine.move(to: PitchSmallCenter)
54                  PitchSmallLine.addLine(to: CGPoint(x:  bounds.width / 2 + ...
        (x2*CGFloat(cos(roll))-y*CGFloat(sin(roll))), y: bounds.height/2 - ...
        (y*CGFloat(cos(roll)) + x2*CGFloat(sin(roll)))))
55              }
56          }
57          UIColor.white.setStroke()
58          PitchSmallLine.lineWidth = 2
59          PitchSmallLine.stroke()
60          //Left Wing
61          let LeftWing = UIBezierPath()
62          let centerAE = CGPoint(x: bounds.width/2 - CGFloat(WingStart), y: ...
        bounds.height / 2 - CGFloat(WingWidth/2))
63          LeftWing.move(to: centerAE)
64          LeftWing.addLine(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(WingSize+WingStart), y: bounds.height / 2 - CGFloat(WingWidth/2)))
65          LeftWing.addLine(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(WingSize+WingStart), y: bounds.height / 2 + CGFloat(WingWidth/2) ))
66          LeftWing.addLine(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(WingStart+WingWidth), y: bounds.height / 2 + CGFloat(WingWidth/2) ))
67          LeftWing.addLine(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(WingStart+WingWidth), y: bounds.height / 2 + CGFloat(WingWidth/2) + ...
        CGFloat(WingSize/4) ))
68          LeftWing.addLine(to: CGPoint(x: bounds.width/2 - CGFloat(WingStart), y: ...
        bounds.height / 2 + CGFloat(WingWidth/2) + CGFloat(WingSize/4) ))
69          LeftWing.close()
70          UIColor.white.setStroke()
71          LeftWing.lineWidth = 5
72          LeftWing.stroke()
73          UIColor.black.setFill()
74          LeftWing.fill()
75          //Right Wing
76          let RightWing = UIBezierPath()
77          let centerAD = CGPoint(x: bounds.width/2 + CGFloat(WingStart), y: ...
        bounds.height / 2 - CGFloat(WingWidth/2))
```

```
78          RightWing.move(to: centerAD)
79          RightWing.addLine(to: CGPoint(x: bounds.width/2 + ...
        CGFloat(WingSize+WingStart), y: bounds.height / 2 - CGFloat(WingWidth/2)))
80          RightWing.addLine(to: CGPoint(x: bounds.width/2 + ...
        CGFloat(WingSize+WingStart), y: bounds.height / 2 + CGFloat(WingWidth/2) ))
81          RightWing.addLine(to: CGPoint(x: bounds.width/2 + ...
        CGFloat(WingStart+WingWidth), y: bounds.height / 2 + CGFloat(WingWidth/2) ))
82          RightWing.addLine(to: CGPoint(x: bounds.width/2 + ...
        CGFloat(WingStart+WingWidth), y: bounds.height / 2 + CGFloat(WingWidth/2) + ...
        CGFloat(WingSize/4) ))
83          RightWing.addLine(to: CGPoint(x: bounds.width/2 + CGFloat(WingStart), y: ...
        bounds.height / 2 + CGFloat(WingWidth/2) + CGFloat(WingSize/4) ))
84          RightWing.close()
85          UIColor.white.setStroke()
86          RightWing.lineWidth = 5
87          RightWing.stroke()
88          UIColor.black.setFill()
89          RightWing.fill()
90          //Center Point
91          let Point = UIBezierPath()
92          let centerPoint = CGPoint(x: bounds.width/2 - CGFloat(WingWidth/2), y: ...
        bounds.height / 2 - CGFloat(WingWidth/2))
93          Point.move(to: centerPoint)
94          Point.addLine(to: CGPoint(x: bounds.width/2 + CGFloat(WingWidth/2), y: ...
        bounds.height / 2 - CGFloat(WingWidth/2)))
95          Point.addLine(to: CGPoint(x: bounds.width/2 + CGFloat(WingWidth/2), y: ...
        bounds.height / 2 + CGFloat(WingWidth/2)))
96          Point.addLine(to: CGPoint(x: bounds.width/2 - CGFloat(WingWidth/2), y: ...
        bounds.height / 2 + CGFloat(WingWidth/2)))
97          Point.close()
98          UIColor.white.setStroke()
99          Point.lineWidth = 5
100         Point.stroke()
101         UIColor.black.setFill()
102         Point.fill()
103         //Arc drawing
104         let Arc = UIBezierPath(arcCenter: CGPoint(x: bounds.width/2, y: ...
        bounds.height / 2), radius: CGFloat(Double(bounds.height)*ArcFactor),
105                                startAngle: CGFloat(-5*pi/6-roll), endAngle: ...
        CGFloat((-pi)/6-roll),
106                                clockwise: true)
107         UIColor.white.setStroke()
108         Arc.lineWidth = 5
109         Arc.stroke()
110         //Big lines
111         let BigLine = UIBezierPath()
112         for i in stride(from: -pi/3, through: pi/3, by: pi/6) {
113             BigLine.move(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(Double(bounds.height)*ArcFactor) * CGFloat(sin(roll + i)), y: ...
        bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) * CGFloat(cos(roll ...
        + i))))
114             BigLine.addLine(to: CGPoint(x: bounds.width/2 - ...
        (CGFloat(Double(bounds.height)*ArcFactor) + CGFloat(PointSize)) * ...
        CGFloat(sin(roll + i)), y: bounds.height/2 - ...
        (CGFloat(Double(bounds.height)*ArcFactor) + CGFloat(PointSize)) * ...
        CGFloat(cos(roll+i))))
115         }
116         BigLine.lineWidth = 5
117         BigLine.stroke()
```

```
118             //Small lines
119             let SmallLine = UIBezierPath()
120             for i in stride(from: -pi/3, through: pi/3, by: pi/18) {
121                 SmallLine.move(to: CGPoint(x: bounds.width/2 - ...
        CGFloat(Double(bounds.height)*ArcFactor) * CGFloat(sin(roll + i)), y: ...
        bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) * CGFloat(cos(roll ...
        + i))))
122                 SmallLine.addLine(to: CGPoint(x: bounds.width/2 - ...
        (CGFloat(Double(bounds.height)*ArcFactor) + CGFloat(PointSize/3)) * ...
        CGFloat(sin(roll + i)), y: bounds.height/2 - ...
        (CGFloat(Double(bounds.height)*ArcFactor) + CGFloat(PointSize/3)) * ...
        CGFloat(cos(roll+i))))
123             }
124             SmallLine.lineWidth = 5
125             SmallLine.stroke()
126             //Magenta Bars
127             let VerticalLine = UIBezierPath()
128             let HorizontalPosition = CGPoint(x: bounds.width/2 + ...
        bounds.width/3/CGFloat(MaxRoll)*CGFloat(CorreccioRoll), y: bounds.height / 6)
129             VerticalLine.move(to: HorizontalPosition)
130             VerticalLine.addLine(to: CGPoint(x: bounds.width/2 + ...
        bounds.width/3/CGFloat(MaxRoll)*CGFloat(CorreccioRoll), y: bounds.height/6 * 5))
131             VerticalLine.lineWidth = 10
132             UIColor.magenta.setStroke()
133             VerticalLine.stroke()
134             let HorizontalLine = UIBezierPath()
135             let VerticalPosition = CGPoint(x: bounds.width/6, y: bounds.height/2 - ...
        bounds.height/3/CGFloat(MaxPitch)*CGFloat(CorreccioPitch) )
136             HorizontalLine.move(to: VerticalPosition)
137             HorizontalLine.addLine(to: CGPoint(x: bounds.width/6*5, y: ...
        bounds.height/2 - bounds.height/3/CGFloat(MaxPitch)*CGFloat(CorreccioPitch)))
138             HorizontalLine.lineWidth = 10
139             UIColor.magenta.setStroke()
140             HorizontalLine.stroke()
141             //Roll Triangle
142             let Triangle = UIBezierPath()
143             Triangle.move(to: CGPoint(x: bounds.width/2, y: bounds.height/2 - ...
        CGFloat(Double(bounds.height)*ArcFactor)))
144             Triangle.addLine(to: CGPoint(x: bounds.width/2 + CGFloat(TriangleSize/2), ...
        y: bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) + ...
        CGFloat(TriangleSize)))
145             Triangle.addLine(to: CGPoint(x: bounds.width/2 + CGFloat(TriangleSize/2), ...
        y: bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) + ...
        2*CGFloat(TriangleSize)))
146             Triangle.addLine(to: CGPoint(x: bounds.width/2 - CGFloat(TriangleSize/2), ...
        y: bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) + ...
        2*CGFloat(TriangleSize)))
147             Triangle.addLine(to: CGPoint(x: bounds.width/2 - CGFloat(TriangleSize/2), ...
        y: bounds.height/2 - CGFloat(Double(bounds.height)*ArcFactor) + ...
        CGFloat(TriangleSize)))
148             Triangle.close()
149             UIColor.white.setStroke()
150             Triangle.lineWidth = 5
151             Triangle.stroke()
152             UIColor.white.setFill()
153             Triangle.fill()
154         }
155     }
```

B.2 Control action file

```
//
//  ViewController.swift
//  ArtificialHorizon
//
//  Created by Guillem Olivella i Mart on 16/11/2018.
//  Copyright 2018 Guillem Olivella. All rights reserved.
//

import UIKit
import CoreMotion
import CoreLocation

//Model variables
var incT = 0.5
//Attitude variables
var pitch = 0.1
var roll = 0.1
var yaw = 0.1
//Kinematic variables
var course = 0.0
var altitude = 0.0
var PzA = 0.0
var PzV = 0.0
var PzI = 0.0
var latitude = 0.0
var longitude = 0.0
var altitudePressure = 0.0
//Magenta Bars
let MaxRoll = 22.0
var CorreccioRoll = 0.0
var ObjectiuRoll = 12.0
let MaxPitch = 12.0
var CorreccioPitch = 0.0
var ObjectiuPitch = 5.0
// Waypoint Selection
var NextWaypointLatitude = 41.479663
var NextWaypointLongitude = 1.929094
var NextWaypointAltitude = 3000.0 //ft
var LatitudeList = [41.43, 41.51, 41.50]
var LongitudeList = [1.79, 1.69, 1.85]
var AltitudeList = [3000.0, 3500.0, 3000.0] //ft
//PID parameters
let Kp = 0.9
let KpA = 0.004
let KiA = 0.003
let KdA = -0.008
let N = 0.1
var K = 0.0
//frac, mu and theta not displayed for its large content.
let frac = [...]
let mu = [...]
let theta = [...]
//Other
let pi = 3.14159
```

```
55 class ViewController: UIViewController,CLLocationManagerDelegate {
56     @IBOutlet weak var Drawing: ArtificialHorizon!
57     @IBOutlet weak var TextPitch: UITextField!
58     @IBOutlet weak var TextRoll: UILabel!
59     @IBOutlet weak var TextCourse: UITextField!
60     @IBOutlet weak var TextAltitude: UITextField!
61     let motionManager = CMMotionManager()
62     let altimeter = CMAltimeter()
63     var timer: Timer!
64     var timer2: Timer!
65     var locationManager = CLLocationManager()
66     override func viewDidLoad() {
67         super.viewDidLoad()
68         //Location Services authorization
69         locationManager.requestAlwaysAuthorization()
70         motionManager.startDeviceMotionUpdates()
71         if CLLocationManager.locationServicesEnabled() {
72             timer = Timer.scheduledTimer(timeInterval: 0.1, target: self, ...
       selector: #selector(ViewController.update), userInfo: nil, repeats: true)
73             timer2 = Timer.scheduledTimer(timeInterval: incT, target: self, ...
       selector: #selector(ViewController.PositionCalculation), userInfo: nil, ...
       repeats: true)
74         }
75         locationManager.desiredAccuracy = kCLLocationAccuracyBestForNavigation;
76     }
77     @objc func update() {
78         if let deviceMotion = motionManager.deviceMotion {
79             roll = deviceMotion.attitude.pitch
80             pitch = deviceMotion.attitude.roll
81             yaw = deviceMotion.attitude.yaw
82             TextPitch.font = UIFont(name:"Courier-Bold", size: 20.0)
83             TextRoll.font = UIFont(name:"Courier-Bold", size: 20.0)
84             TextPitch.text = ("PITCH: \(Int(-pitch/pi*180-90))")
85             TextRoll.text = ("\(Int((roll*180/pi)))")
86         }
87         determineMyCurrentLocation()
88         TextCourse.font = UIFont(name:"Courier-Bold", size: 20.0)
89         TextAltitude.font = UIFont(name:"Courier-Bold", size: 20.0)
90         TextCourse.text = ("COURSE: \(Int(course))")
91         TextAltitude.text = ("ALTITUDE: \(Int(altitude)) m")
92         Drawing.setNeedsDisplay()
93 }
94             func determineMyCurrentLocation() {
95         locationManager = CLLocationManager()
96         locationManager.delegate = self
97         locationManager.desiredAccuracy = kCLLocationAccuracyBest
98         locationManager.requestAlwaysAuthorization()
99         if CLLocationManager.locationServicesEnabled() {
100             locationManager.startUpdatingLocation()
101             //locationManager.startUpdatingHeading()
102         } else {
103             print("GPS not available")
104         }
105     }
106     func locationManager(_ manager: CLLocationManager, didUpdateLocations ...
       locations: [CLLocation]) {
107         let userLocation:CLLocation = locations[0] as CLLocation
108         course = userLocation.course
109         altitude = userLocation.altitude
```

```
110         latitude = userLocation.coordinate.latitude
111         longitude = userLocation.coordinate.longitude
112     }
113
114     func locationManager(_ manager: CLLocationManager, didFailWithError error: Error)
115     {
116         print("Error \(error)")
117     }
118     @objc func PositionCalculation (){
119         var NBA = atan2 (cos(NextWaypointLatitude/180*pi)*sin(longitude/180*pi - ...
        NextWaypointLongitude/180*pi),(cos( ...
        latitude/180*pi)*sin(NextWaypointLatitude/180*pi))-(sin(latitude/180*pi)*cos ...
        (NextWaypointLatitude/180*pi)*cos (longitude/180*pi - ...
        NextWaypointLongitude/180*pi)))/pi*180
120         if (NBA<0){
121             NBA = 360 + NBA
122         }
123         var zeta = NBA - course
124         if (zeta > 180){
125             zeta = zeta - 360
126         }
127         if (zeta < -180){
128             zeta = zeta + 360
129         }
130         let Pz = NextWaypointAltitude*0.3048 - altitude
131         PzV = (Pz - PzA)/incT
132         PzA = Pz
133         if Pz<0 {
134             if zeta > 20 {
135                 zeta = 20
136             } else if zeta < -20 {
137                 zeta = -20
138             }
139            RollTarget = Kp * zeta;
140             if Pz < -11 {
141             PitchTarget = -2.5;
142             } else {
143                 PitchTarget = (KpA * Pz + KdA * PzV * N + KiA * PzI)/pi*180;
144                 if PitchTarget < -2.5 {
145                     PitchTarget = -2.5
146                 } else if PitchTarget > 8 {
147                     PitchTarget = 8
148                 } else {
149                     PzI = PzI + Pz*incT
150                 }
151             }
152         } else {
153             if zeta != 0 {
154                 K = Pz/(zeta/180*pi)
155                 if K < -2000 {
156                     K = -2000
157                 } else if K > 2000 {
158                     K = 2000
159                 } else if K < 4 && K > 0 {
160                     K = 4
161                 } else if K > -4 && K ≤ 0 {
162                     K = -4
163                 }
164                 if K>0{
```

```
                    var i = 0
                    while K < frac[i] {
                        i+=1
                    }
                    PitchTarget = (KpA * Pz + KdA * PzV * N + KiA * PzI)/pi*180;
                    if PitchTarget < theta[i] {
                        if PitchTarget < −2.5 {
                            PitchTarget = −2.5
                        } else if PitchTarget > 8 {
                            PitchTarget = 8
                        } else {
                            PzI = PzI + Pz*incT
                        }
                    } else {
                        PitchTarget = theta[i];
                    }
                    if zeta*Kp < mu[i] {
                        RollTarget = zeta*Kp;
                    } else {
                        RollTarget = mu[i];
                    }
                }
                if K<0{
                    K = K * −1
                    var i = 0
                    while K < frac[i] {
                        i+=1
                    }
                    PitchTarget = (KpA * Pz + KdA * PzV * N + KiA * PzI)/pi*180;
                    if PitchTarget < theta[i] {
                        if PitchTarget < −2.5 {
                            PitchTarget = −2.5
                        } else if PitchTarget > 8 {
                            PitchTarget = 8
                        } else {
                            PzI = PzI + Pz*incT
                        }
                    } else {
                        PitchTarget = theta[i];
                    }
                    if zeta * −Kp < mu[i] {
                        RollTarget = zeta * −Kp
                    } else {
                        RollTarget = −mu[i]
                    }
                }
            } else {
                RollTarget = 0
                PitchTarget = (KpA * Pz + KdA * PzV * N + KiA * PzI)/pi*180;
                if PitchTarget < −2.5 {
                    PitchTarget = −2.5
                } else if PitchTarget > 8 {
                    PitchTarget = 8
                } else {
                    PzI = PzI + Pz*incT
                }
            }
        }
        RollCorrection = Rolltarget − roll/pi*180;
```

```
224          if (RollCorrection > MaxRoll ){
225              RollCorrection = MaxRoll
226          } else if (RollCorrection < MaxRoll * -1 ){
227              RollCorrection = MaxRoll * -1
228          }
229          PitchCorrection = PitchTarget + pitch/pi*180 + 90;
230          if (PitchCorrection > MaxPitch ){
231              PitchCorrection = MaxPitch
232          } else if (PitchCorrection < MaxPitch * -1 ){
233              PitchCorrection = MaxPitch * -1
234          }
235                       let A1 = (sin(latitude - ...
        NextWaypointLatitude)/2*pi/180)*(sin(latitude - NextWaypointLatitude)/2*pi/180);
236          let B1 = cos ...
        (latitude*pi/180)*cos(NextWaypointLatitude*pi/180)*sin((longitude - ...
        NextWaypointLongitude)/2*pi/180)*sin((longitude - ...
        NextWaypointLongitude)/2*pi/180);
237          let d = 6371000 * 2 * asin (sqrt(A1+B1));
238          if (d<1000){
239              if (cont < 3){
240                  cont+=1;
241                  NextWaypointLatitude = LatitudeList[cont];
242                  NextWaypointLongitude = LongitudeList[cont];
243                  NextWaypointAltitude = AltitudeList[cont];
244              }
245              if (cont ≥ 3){
246                  cont = 0;
247                  NextWaypointLatitude = LatitudeList[cont];
248                  NextWaypointLongitude = LongitudeList[cont];
249                  NextWaypointAltitude = AlitudeList[cont];
250              }
251          }
252      }
253  }
```